LARGE-SCALE EVOLUTIONARY OPTIMIZATION USING MULTI-LAYER STRATEGY DIFFERENTIAL EVOLUTION

By: Tarik Eltaeib

Under the Supervision of Prof. Ausif Mahmood

DISSERTATION

SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIRMENTS

FOR THE DEGREE OF DOCTOR OF PHILOSOHPY IN COMPUTER SCIENCE

AND ENGINEERING

THE SCHOOL OF ENGINEERING

UNIVERSITY OF BRIDGEPORT

CONNECTICUT

May 2019

LARGE-SCALE EVOLUTIONARY OPTIMIZATION USING MULTI-LAYER STRATEGIES DIFFERENTIAL EVOLUTION

Tarik Eltaeib

Under the Supervision of Dr. Ausif Mahmood

Approvals

Committee Members

Name	Signature	Date
Dr. Ausif Mahmood	Ausif Mahmood	5-13-2019
Dr. Prabir Patra	P. Patra	5/13/19
Dr. Miad Faezipour	Miad Faezipour	5,13,2019
Dr. Abdel-shakour Abuzneid	AS Abuzneid	5-13-2019
Dr. Emre Tokgoz	Emre T.	5/13/19

Ph.D. Program Coordinator

Dr. Khaled M. Elleithy — 5/23/19

Chairman, Computer Science and Engineering Department

Dr. Ausif Mahmood — 5-13-2019

Dean, School of Engineering

Dr. Tarek M. Sobh — 5/15/2019

LARGE-SCALE EVOLUTIONARY OPTIMIZATION USING MULTI-LAYER STRATEGY DIFFERENTIAL EVOLUTION

LARGE-SCALE EVOLUTIONARY OPTIMIZATION USING MULTI-LAYER STRATEGY DIFFERENTIAL EVOLUTION

ABSTRACT

Differential evolution (DE) has been extensively used in optimization studies since its development in 1995 because of its reputation as an effective global optimizer. DE is a population-based metaheuristic technique that develops numerical vectors to solve optimization problems. DE strategies have a significant impact on DE performance and play a vital role in achieving stochastic global optimization. However, DE is highly dependent on the control parameters involved. In practice, the fine-tuning of these parameters is not always easy. Here, we discuss the improvements and developments that have been made to DE algorithms.

The Multi-Layer Strategies Differential Evolution (MLSDE) algorithm, which finds optimal solutions for large scale problems. To solve large scale problems were grouped different strategies together and applied them to date set. Furthermore, these strategies were applied to selected vectors to strengthen the exploration ability of the algorithm. Extensive computational analysis was also carried out to evaluate the performance of the proposed algorithm on a set of well-known CEC 2015 benchmark

functions. This benchmark was utilized for the assessment and performance evaluation of the proposed algorithm.

ACKNOWLEDGEMENTS

Special appreciations to my family. Words cannot express how thankful I am to my wife, my daughters, my mother, my mother-in law, and father-in-law for all of the sacrifices and prayers for me that you've made on my behalf. I would like to express my special thanks to my advisor Professor Ausif Mahmood and his efforts, encouraging and guiding me to grow as a research scientist.

.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1: INTRODUCTION

1.1 Background

Optimization algorithms are important approaches for resolving hard optimization problems [1]. Optimization is defined as the procedure of discovery that provides the minimum or maximum value of a function f(x) [2, 3]. There are many reasons that make this problem difficult to solve. First, we cannot perform a comprehensive search if the problem domain space is too large. Second, the evaluation function is noisy or varies with time, generating a series of solutions instead of a single solution. Third, sometimes the constraints prevent arriving at a possible solution such that the optimization approach is the only solution [4].

Differential evolution (DE) is a stochastic algorithm for solving numerical continuous optimization problems. Since its inception, the DE algorithm has been a powerful global optimizer. DE was developed by Kenneth Price in 1994 and has since become a promising optimization algorithm that converges to the real optimum without using significant amounts of resources. Furthermore, its performance was validated in the evolutionary domain by the IEEE Conference on Evolutionary in 1996 [5].

More recently, different versions of DE have secured the top ranks in many competitions between evolutionary algorithms (EAs) by the IEEE Congress on

Evolutionary Computation (CEC) conference series (http://www.ntu.edu.sg/home/epnsugan/index_files/cec- benchmarking.htm).

1.2. Research Problem and Scope

An impressive number of different DE algorithms have been introduced by the research community over the past decades because various DE algorithms involving different techniques. To differentiate among those techniques, we need to define a comprehensive framework that helps to deepen understanding of the characteristics of different DE strategies, with the goal of benefiting from the various approaches. In fact, understanding how to combine these DEs harmoniously and their underlying concepts could be crucial to attaining effective designs or improving the performance of DE algorithms in particular or any optimization algorithms in general. Moreover, the literature shows that no single algorithm has been demonstrated to be effective for various applications.

DE algorithms are different from EA algorithms that shape offspring by mixing solutions with a difference factor rate of selected individual vectors and are an alternative to recombining individuals through a probabilistic scheme. In fact, the differential mutation strategy is the main component that distinguishes DE from other population algorithms. Applying the mutation to all candidates defines an exploration rule based on other candidate solutions. Therefore, the mutation strategy enhances a population's capability for discovering new promising offspring based on the current distribution of solutions within the domain space. Ideally, the performance of DE is based on two major

components: the chosen strategy and the control parameters. However, the strategy underlying DE consists of mutation, crossover and selection operators, which are utilized at each generation to determine the global optimum. The control parameter components consist of the population size NP, scaling factor F and the crossover rate Cr.

1.3. Motivation behind the Research

Despite the potential of DE, it is obvious to the research community that some adjustments to classic DE are essential to significantly enhance its performance, especially in addressing high-dimension problems. Stagnation, premature, convergence, and sensitivity are the control parameters that influence the performance of DE. To evaluate the reliability and robustness of the different DE algorithms, we introduce a general framework that includes the control parameters for evaluating the efficiency of the different algorithms. For example, stagnation occurs when the population cannot converge to a suboptimal solution although the diversity of the population remains high. This does not improve the population over a period of iterations, and the algorithm is not capable of finding a new search domain. There are many causes of stagnation, including control parameters that become inefficient for a specific problem in the decision space. Many studies have proposed a variety of ways to improve the current DE algorithm through modifications, including the use of differential mutations with perturbations, mutations with selection pressure, and operator adaption techniques. To address this need, we have conducted an extensive study on differential evolution and observed that

the performance of differential evolution and the quality of the results are based on the type of technique used, and what control parameters are effective.

1.4. Potential Contribution of the Proposed Research

We propose a Multi-Layer Strategies Differential Evolution (MLDE) approach, which uses different mutation strategies in order to reach a fast convergence rate and avoid premature convergence due to the loss of diversity in the population. In fact, we used multilayer crossover techniques since there no single method has proven fit for every problem; though a crossover scheme may work perfectly with some problems, they may perform poorly with others. Each problem has different characteristics: some research showed that a scheme such as binomial crossover performed well with some type of problem. MLDE works to improve the diversification of offspring by using different strategies in a multiple-layer approach. This approach makes the population widely spread so the sampled vectors can easily generate better new offspring. This technique can accelerate convergence rate for finding the optimum solution with a smaller number of iterations.

Another significant factor is considered in this work is to provide a comprehensive study of the different types of state-of-the-art differential evolution algorithms that are available as global numerical optimizations in continuous search space. This comprehensive study sheds light on most improvements and developments pertaining to different types of DE families, including primary concepts and a variety of DE formats.

1.5 Literature Survey and Background

1.5.1 Classic Differential Evolution

If we are seeking the optimum for X* which demonstrate by vector X_i^*,i=1,…D , X $\in \mathbb{R}^D$, within boundary constraints L ≤ X ≤ H. Differential evolution (DE) is population-based, where the initial population $P_{i,j,} \in \mathbb{R}^{[D \times NP]}$ with random initialization $H \leq P_{i,j,} \leq L$. Initialization of the population is important step that assuming that there is no previous information about the optimum solution. Therefore, the population is initialized within only boundary constraints upper bound (H) and lower bound (L), so the population can by initialized by the following

$$P_{i,j} = L + (H - L).rand_{i,j}[0,1)\ , i = 1,2, \ldots., D\ ; j = 1,2, \ldots., NP$$

After the initialization phase, the evolution involves the three processes of mutation, crossover, and selection. The classic differential evolution strategy consists of three random vectors v_1, v_2, and v_3 that are selected from the population (Eq. 1). Randomly select of three individuals from the population

$$v_{1,2,3} \in [1, \ldots. NP]\ , where\ v_1 \neq v_2 \neq v_3$$
$$v_1 = (int)\ (rand(\ ) * NP\ );$$
$$v_2 = (int)\ (rand(\ ) * NP\ );$$
$$while\ (v_2 = v_1)\ v_2 = (int)\ (rand(\ ) * NP\ );$$
$$v_3 = (int)\ (rand(\ ) * NP\ );$$
$$\text{while } (v_2 = v_1 \,||\, v_2 = v_3\ )\ v_3 = (int)\ (rand(\ ) * NP\ )$$

The mutation operation recombines to construct the mutation vector v_y shown in Figure 1. The associated equation.

$$v_y = v_3 + F * (v_1 - v_2) \quad (1)$$

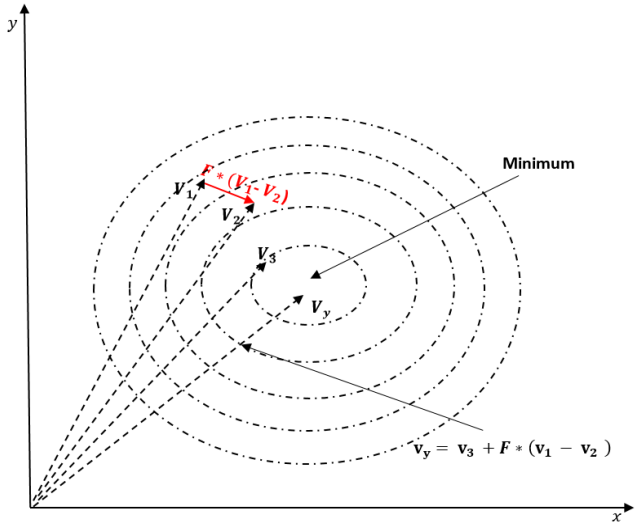


Figure **1.1** Random vectors selected in the mutation strategy (classic DE)

The mutation process is the main distinctive component of DE and is considered the strategy by which DE is carried out. There are different types of mutation strategies, each one distinguished with an abbreviation based on the classic mutation strategy described by equation (1), i.e., DE/rand/1/bin, where DE represents differential evolution and "rand" represents random, which indicates that the vectors are selected randomly. The number one indicates the number of difference pairs; in this strategy, it is one pair$(v_1 - v_2)$. The last term represents the type of crossover used. This term could be "exp," for exponential, or "bin," for binomial [9]. Then, to complement the previous step (mutation strategy), DE also apply uniform crossover to construct trial vectors v_{Trail} which is out of parameter values that have been copied from two different vectors. In particular, DE selected random vector from population indicate as v_x which must be different of v_1, v_2, and v_3; and then it crosses with a mutant vector v_y ; the binomial crossover is generated as follows:

$$v_{Trail}[i,j] = \begin{cases} v_y[i,j] & if\ rand(0,1) < Cr \\ v_X[i,j] & otherwise \end{cases} \quad (1.2)$$

The crossover probability, Cr ∈ (0, 1), is a pre-defined rate that specify the fraction of parameter that are transferred from the mutant. Thus, it use to control which source participate a given parameter. Uniform crossover rate compares with uniform random values form from rand (0,1); if the random value is smaller than or equal to Cr then the trial parameter is copied from mutant vector v_y else the parameter is inherited from v_X

The next operation is selection, in which the trail vector v_{Trail} competes with the target vector v_x . If this trail vector v_{Trail} is equal or less than v_x it changes the target vector v_x in the next generation else v_x not changed in the population

$$v_x = \begin{cases} v_{Trail} & if\ f(v_{Trail}) \le f(v_X) \\ v_X & otherwise \end{cases}$$

Where $f(x)$ is the objective function? Therefore, if the new trail vector $f(v_{Trail})$ is less than or equal to the target vector $f(v_X)$, it replaces the target vector. Otherwise, the population maintains the target vector value. Therefore, the different DE phases prevent the population from ever deteriorating; the population either remains the same or improves. Furthermore, continued refining of the population is updated by the trial vector, although the fitness of the trial vector is the same as that of the current vector. This factor is crucial in DE because it provides the algorithm the ability to move through the landscape using a variety of generations [10]. The termination condition can be either a preset maximum number of generations or a pre-specified target of the objective function value. [11].

1.5.2 Differential Evolution Strategies

Table (1.1) The differentiation operation can be carried out using many mutation strategies.

	Strategy	Formulation
1.	DE/best/1/exp	$V_{(G+1)} = V_{(best,G)} + F.[V_{(1,G)} - V_{(2,G)}]$
2.	DE/rand-to-best/1/exp	$V_{(G+1)} = V_{(i,G)} + \lambda.[V_{(1,G)} - V_{(2,G)}] + F.[V_{(1,G)} - V_{(2,G)}]$
3.	DE/best/2/exp	$V_{(G+1)} = V_{(best,G)} + F[V_{(1,G)} + V_{(2,G)} - V_{(3,G)} - V_{(4,G)}]$
4.	DE/rand/2/exp	$V_{(G+1)} = V_{(5,G)} + F[V_{(1,G)} + V_{(2,G)} - V_{(3,G)} - V_{(4,G)}]$
5.	DE/best/1/bin	$V_{(G+1)} = V_{(best,G)} + F.[V_{(2,G)} - V_{(3,G)}]$
6.	DE/rand/1/bin	$V_{(G+1)} = V_{(1,G)} + F.[V_{(2,G)} - V_{(3,G)}]$
7.	DE/rand-to-best/1/bin	$V_{(G+1)} = V_{(i,G)} + \lambda.[V_{(best,G)} - V_{(i,G)}] + F.[V_{(1,G)} - V_{(2,G)}]$
8.	DE/best/2/bin	$V_{(G+1)} = V_{(best,G)} + F[V_{(1,G)} + V_{(2,G)} - V_{(3,G)} - V_{(4,G)}]$
9.	DE/rand/2/bin	$V_{(G+1)} = V_{(5,G)} + F[V_{(1,G)} + V_{(2,G)} - V_{(3,G)} - V_{(4,G)}]$

The various equations underpinning DE possess certain aspects in common when applied for continuous optimization. All consist of an original point sometimes referred to as the base point. The original algorithm carries out the search operation such that it finds the optimum as soon as possible. We can generalize the DE formula to the form $\alpha = \beta + F \cdot \delta$, where β represents the base vectors and δ the difference between vectors. Thus, the main goal of all DE equations is to provide the optimal direction based on the differential β and base vector δ (Figure 2).

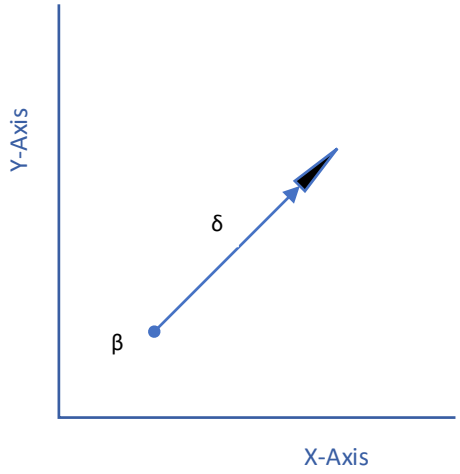


Figure 1.2 The differential β and base vector δ provide the optimal direction

Establishing β and δ is crucial to creating an efficient strategy that can be applied to the chosen individuals from the population. However, all possible combinations of β and δ can be classified into the following strategies: local, random, directed, and hybrid. In random strategies, abbreviated as "Rand", all individuals are formed randomly, and there is no prior information about the objective function. In directed strategies, abbreviated as "DIR", a suitable value for the base vector is chosen according to the objective function to ensure a suitable direction. Hybrid strategies include the combination of "Rand" and "DIR", labeled RAND/DIR. In another approach, the best overall vector is used, not only the best among the selected individuals; this approach is referred to as the "BEST". Combining the "Rand" and "BEST" yields the hybrid RAND/BEST strategy. In addition, the combination of more than two approaches, e.g., RAND/BEST/DIR, can yield favorable results by exploiting the advantages of each approach.

However, Table 3.1 shows that all DE strategies employed are formed based one the DE/rand/x variation, which applies pairs of difference vectors:

$$u_i = x_{r_1} + F_1 (x_{r_1} - x_{r_1}) + \cdots \ldots \ldots + F_k (x_{r_{2k}} - x_{r_{2k}})$$

whereas the scaling factors are frequently presumed to be the same F1 = F2 =⋯ = Fk = F. Substituting an arbitrary base vector **x**1 as vbest, "the best vector" from the population, provides a different DE approach, indicated DE/best/1:

$$\mathrm{v_y} = \mathrm{v_{best}} + F * (\mathrm{v_1} - \mathrm{v_2})$$

Most mutation strategies can be formed by a general formula based on the sum of k scaled difference vectors and a weighted average among the best vector and arbitrary ones:

$$\mathrm{v}_y = \lambda\, \mathrm{v_{best}} + (1 - \lambda)\, \mathrm{v_x} + \sum F_i * (\mathrm{v}_{r2_i} - \mathrm{v}_{r2_{i+1}})$$

One aspect common to all the mutation strategy methods is the base vector, which controls the search direction. The difference vector provides a mutation rate term, such as a self-adaptive term, that is added to an arbitrary or guided base vector to construct a trial individual. Over generations, the individuals of a population reside in increasingly better positions and reform themselves. The various combinations of these vectors can be categorized into four groups based on information pertaining to the values gathered from the objective function: random, directed, local and hybrid.

The RAND approach consists of strategies in which the trial individual is produced without knowledge of the value of the objective function. Similarly, the RAND/DIR approach includes strategies that use the values of the objective function to determine a promising direction. Likewise, the RAND/BEST approach applies the best

individual approach to proceed with a trial. Additionally, the RAND/BEST/DIR approach combines the last two groups into one that includes all of their collective benefits.

However, a suitable direction is obtained by using the best individual to decrease the search space and exploration time [12, 13]. Thus, the “dir” and “dir-best” strategies, which use objective function values to generate trial individuals, can produce an exploitation function. In fact, the random selection of parents for a trial enhances exploration capabilities [14-16]. Thus, the locations of individuals carry information about the fitness landscape. Therefore, an effective mutation strategy that leads to uniform random vectors represents the entire search space well.

1.5.3 Initialization

DE is a population-based optimization technique that begins with the problem solution by selecting the objective function at a random initial population. Predefined parameter bounds describe the area from which the number of population (Np) vectors in this initial population is chosen within both the upper bound “b_U” and the lower bound “b_L”, where the subscripts L and U indicate lower and upper, respectively. The following equation is used to develop a random number generator for all vectors from within the predefined upper and lower bounds. The random function Random(0, 1) outputs a uniform random number within the range (0, 1).

$$x_i = Random(0,1).(b_U - b_L)+ b_L$$

1.5.4 Crossover

To balance the differential mutation search strategy, DE also applies uniform crossover to construct trial vectors. A trial vector is constructed from values that have been copied from two diverse vectors. In particular, DE crosses each vector as follows:

$$u_{i,g} = u_{j,i,g} = \begin{cases} v_{j,i,g} \text{ If } (rand_j(0,1) \leq Cr \text{ or } j = j_{rand} \\ x_{j,i,g} \end{cases}$$

The crossover probability, $\backsim r \in [0,1]$, is predefined in the classic version of DE, and the fraction value of the Cr control is cloned from the mutant vector. C_r is compared with a random number randj(0,1). If the random number is less than or equal to C_r, the trial parameter is inherited from the mutant $v_{j,i,g}$; otherwise, the parameter is cloned from the vector $x_{j,i,g}$.

1.5.5 Selection

In this stage, we determine when the trial vector $u_{i,g}$ has an objective function value that is less than or equal to that of its target vector $x_{i,g}$. DE swaps the target vector in the next iteration; otherwise, the target retains its place in the population. This process is carried out by comparing each trial vector with the target vector from which the parameters are cloned. After the population is updated, mutation, recombination and selection are repeated until the optimum value is found or after a predefined stop criterion is reached, such as a certain number of iterations.

$$x_{i,g+1} = \begin{cases} u_{i,g} \text{ If } (f(u_{i,g}) \leq f(x_{i,g}) \\ x_{i,g} \end{cases}$$

1.5.6 DE Applications and related automated

Due to the rapid rise of DE as a modest and strong optimizer, developers have applied the technique in a wide range of domains and fields of technology1. Yalcin proposed a new method for the 3D tracking of license plates from video using a DE algorithm, which could be fine-tuned according to the license plate boundaries [17]. A color image quantization application using DE was proposed by Qinghua and Hu. The main objective of image processing techniques is to, during the color image quantization phase, decrease the number of colors in an image with a low amount of deformation. DE can be used to adjust colormaps and find the optimal candidate colormap [18]. With respect to the bidding market, Alvaro et al. applied DE in developing a competitive electricity market application that finds the optimal bids based on daily bidding activity [19]. Sickel et al. used DE in developing a power plant control application for a reference governor to produce an optimal group of points for controlling a power plant that was produced by [20]. Wang et al. proposed a flexible QoS multicast routing algorithm for the next-generation Internet that improves the quality of service (QoS) of multicasts to manage the increasing demand of network resources [21]. With respect to the electric power systems industry, Ela et al. applied DE to determine the optimal power flow [22]. Goswami et al. proposed a DE application for model-based well log-data inversion to

[1] http://www1.icsi.berkeley.edu/~storn/code.html

discover features of earth formations based on the dimensions of physical phenomena [23]. Another application applies network system reconfiguration for distributing systems. The network reconfiguration application proposed by Tzong and Lee involves the application of Improved Mixed-Integer Hybrid Differential Evolution [24]. Another DE application developed by Boughari et al. sets suitable controllers for aircraft stability and control augmentation systems [25].

1.5.7 Parameter Control

The DE algorithm is a simple and effective optimization algorithm for problems from real world when its control parameters are suitably set [8, 26, 27], as reviewed in the previous section. In this section, we review the most current improvement approaches for DE. First, the DE algorithm applies certain control parameters to the system implementation. The accomplishment of DE is influenced by the value of parameters, such as the crossover and mutation rate. Although some studies have recommended certain values for these parameters, their effect on performance is complex and their exact values are unclear. In particular, there is a wide variety of different recommended values that are appropriate for different problems [28-30].

The mutation rate "F", crossover rate "C_r" and population dimension "Np" maintain balance between exploration and exploitation [6]. Exploration is associated with finding new solutions, and exploitation is associated with searching for new suitable solutions; the two processes are linked in the evolutionary search [31, 32]. Therefore, the mutation and crossover rates influenced the convergence rate and the effectiveness of the search space [33].

However, specifying suitable values for these rates is not easy [34]. Three types of strategies are used to set these parameter controls: deterministic parameter control (sometimes called random), self-adaptive parameter control and adaptive parameter control [10, 35-37]. Adaptive and self-adaptive parameter control [9-14, 38-41] have recently been proposed to dynamically alter the control parameters without requiring the user's prior knowledge or information about the problem behaviors throughout the search process [42-46]. In the following sections, the self-adaptive parameter, the adaptive parameter, and hybrid control strategies are discussed.

1.5.8 Deterministic Parameter Control

The parameters are altered using a deterministic rule regardless of the feedback from the evolutionary search, with Jitter and Dither being two operators that are used in this technique. Dither scales the distance of the vector differentials as the same factor, F_i, is applied to all the elements of a subtracted vector. Jitter multiplies each vector element of the subtracted vector by a different scale factor, F_j. The rotation creates jitter using an essentially different procedure than the classic DE's constant mutation with F. However, this approach shows robustness for non-deceiving objective functions [3]. Nonetheless ,applied fixed values for each iteration, and F was created for each individual within the range [0.4, 1] range, whereas the interval [0.5, 0.7] was selected for Cr [47, 48].

Another approach is the composite DE (CoDE) algorithm proposed by Wang et al. In CoDE, a trial vector is selected from a set of groups produced by utilizing diverse DE strategies [49]. The main objective is to arbitrarily merge many trial vector strategies with different parameter at each iteration to construct new trial vectors. These

combinations help solve many problems successfully. Wang et al. used group of trial vector strategies and group of control parameter (almost three) to create strategy and parameter candidate pools. The selected strategies are DE/rand/1/bin, DE/rand/2/bin and DE/current-to- rand/1, and the three pair common choices for the control parameter settings were (F= 1.0, Cr =0.1), (F =1.0; Cr=0.9), and (F =0.8; Cr=0.2). In each generation, the three different strategies are applied, which randomly pick any of the control parameter values.

Then, the trial vector is designated the candidate with the better value of fitnes. The parameters are chosen based on whether they are frequently implemented with many DEs, and their performances are evaluated. The three pairs of parameter settings that provide diverse effects produce new improved candidates. Furthermore, the different values of the control parameters maintain different levels of search performance.

1.5.9 Adaptive Parameter Control

The adaptive technique has been applied with classic DE/rand/1/bin; while the performance is relatively favorable, the technique still suffers from convergence rate issues [43, 44]. Very good designed a self-adaptive and adaptive parameter controls can enhance the robustness and the convergence rate by automatic adapting to the parameters. Approaches other than using the best explored solution use minor resolution in previous generations and their variation with the present population as a good area for finding the optimum. Adapting the parameters is a method called the adaptive DE algorithm (ADE), which applies a adapting evaluation from feedback of F relay on additional parameter (ϒ) that necessity be adjusted[50, 51]. However, the self-adaptive parameter controls the

value assignments and adjusts them dynamically. A parameter is altered dynamically through processing according to pre-defined rules using adaptive control, self-adaptive control or a combination thereof [34, 52]. In addition, the self-adaptive parameter control mix explores the optimal parameter values with the goal of finding the optimal solutions. Indeed the adaptive approach have been achieved success with different technologies; Shojafar et al., 2016 applied the adaptive technique within cloud in order to reach the communication optimization framework and exploiting virtualization technologies[53, 54].

The main purpose of adaptive DE is to help exploit and explore relationships that avoid premature convergence problems and to optimize the final results. In general, there are many techniques for hybridizing a conventional evolutionary algorithm to solve optimization problems. The initial population of DE is formed by problem-specific heuristics. Then, other solutions obtained using another EA might be enhanced with a local search. This type of combination is called a memetic algorithm [10, 52]. The benefits of this hybridization lead to various operators that might exploit problem knowledge, such as merging more promising individuals to be inherited. Furthermore mutation operations may be biased to contain solutions of promising individuals with higher probabilities than those of others.

1.5.10 Differential Evolution with Self-Adapting Populations (DESAP)

Differential Evolution with Self-Adapting Populations (DESAP) dynamically adjusts the crossover and mutation parameters δ , η and the population size π [39]. Each individual i is connected to its control δ_i, η_i, and π_i. δ and π have similar meanings to NP

and CR correspondingly. The mutation factor F is retained as static, and η denotes the probability of implementing an extra mutation using normally distributed after crossover. The main technique of DESAP is unlike that of the traditional DE/rand/1/bin algorithm [38]. Parameters are adapted by developing them over the mutation and crossover processes, as the procedures are applied to each x_i. The updated values of that parameters continue with u_i if $f(u_i) < f(x_i)$. However, DESAP still requires further development to produce better performance. In fact, despite its simplicity, DESAP performs better than DE in one of De Jong's five exam problems, whereas the other solutions are very identical. DESAP represents an opportunity to reduce the control parameters further by updating the size of population, as is done with the additional parameters.

1.5.11 Fuzzy Adaptive Differential Evolution (FADE)

Fuzzy adaptive differential evolution (FADE), presented by Lampinen and Liu [41], is a different type of DE algorithm that apply fuzzy logic controllers to adjust the controller parameters F_i and Cr_i for the crossover and mutation operations. Similarly to DESAP, the size of the population is presumed to be adjusted and is static during the evolution procedure [9]. The fuzzy-logic control method has been verified on a group of 10 functions as benchmark and displays best solutions than those of classic DE for high-dimensional problems.

1.5.12 Self-adaptive Differential Evolution (SaDE)

Self-adaptive Differential Evolution (SaDE) is simultaneously applied to pair of mutation techniques "DE/rand/1" and "DE/current-to-best/2" [45]. The adaptation technique of parameter consists of two chunks: the probability of the adaptation p_i, where i = (1, 2), and the DE parameters F_i and Cr_i. The probability of producing a mutation vector based on the two strategies approaches 0.5 and is updated every 50 iterations using the following method:

$$p_1 = \frac{ns_1\,(ns_2 + nf_2)}{ns_2(ns_1 + nf_1) + ns_1(ns_2 + nf_2)}$$

$$p_2 = 1 - p_1$$

where ns_i and nf_i are the numbers of offspring vectors constructed by the i_{th} i = (1, 2) strategy that was a success or failure in the selection process over the last 50 generations. It is assumed that this adaptation process can progressively develop the most appropriate mutation strategy at diverse learning phases for a given problem. The mutation factors F_i are autonomously created at each iteration based on a normal distribution "NR" with a mean of 0.5 and a standard deviation of 0.3,

$$F_i = NR(0.5, 0.3)$$

The crossover rates Cr_i are autonomously formed based on a normal distribution with a mean of Cr_m and a standard deviation of 0.1. The mean Cr_m approaches 0.5, is changed every 25 iterations and is set to be the mean of the effective Cr over the previous 25 generations.

$$Cr_i = NR(Cr_m, 0.1)$$

$$Cr_m = \frac{1}{k}\sum_{k=1}^{k} Cr_{suc}(k)$$

where K is the counters of effective Cr values and $Cr_{suc}(k)$ indicates the k_{th} value.To accelerate the convergence, a local search technique (Quasi–Newton method) is applied to respectable individuals after 200 generations. SaDE has been further developed by applying five mutation strategies to resolve a group of constrained problems [55].

One of the success fuzzy application that applied on cloud computing which consists of numerous of computers linked over instantly transmission network, so it provides the capability to instantaneously perform an numerous software on connected workstations. The job scheduling is one of vital and interesting aspects in cloud computing. FUGE based on fuzzy theory and genetic algorithm that assign jobs to resources optimally considering execution time and cost(memory, virtual machine speed, network rate, and job intervals) .Applied fuzzy theory with modified the standard genetic algorithm (SGA) and used to invention a fuzzy-based steady-state GA. In this approach, jobs are denoted as genes and resources of computing allocated to these genes, and groups of genes produce chromosomes. They created two different forms of chromosomes: first type is based on job length, CPU speed and size of the resources and another form is rely on job length and bandwidth of resources. For each type of chromosome, population of genes are randomly created and computational resources are assigned to gene randomly. Algorithm calculates the value of fitness for each chromosome using a fuzzy function. Fuzzy theory is also used in the crossover step of the GA. In general, single point or two point crossover are used in crossover approach, but in this approach, fuzzy-based crossover that is one of the new approach [56].

1.5.13 Self-adaptive NSDE (SaNSDE)

Neighborhood search differential evolution (NSDE) is similar to classic DE except that Eq. (1) which is replaced with

$$v_y = v_3 + \begin{cases} d_i * N(0.5, 0.5) & if\ u(0.1) < 0.5 \\ d_i * \delta & otherwise \end{cases}$$

where $d_i = v_1 - v_2$ is the differential deviation, N (0.5,0.5) means a Gaussian random number with a average of 0.5 and a standard deviation of 0.5 and δ indicte a Cauchy random variable with a rate parameter of t=1.

Self-adaptive DE (SaDE) [8] was developed to resolve the control parameters and learning technique . In SaDE, two DE learning strategies are chosen according to their performance. The most appropriate learning technique and parameter values are increasingly self-adapted according to the learning experience gained during evolution [57].

SaNSDE is an adaptive differential evolution algorithm that produces mutation vectors in a manner similar to SaDE [57]. However, the difference is that the mutation factors are established based on a normal distribution or a Cauchy distribution:

$$f_i = \begin{cases} Normal\ \text{distribution}(0.5, 0.3), if\ rand < fp \\ \text{Cauchy distribution}\ (0.0, 1.0), otherwise \end{cases}$$

where the normal distribution (μ, σ^2) indicates a random value of mean μ and variance σ^2 and a $\text{Cauchy distribution}$ (μ, δ) indicates a random value with scale parameters μ and δ. The probability fp of the spread over is adapted as follows.

$$p_1 = \frac{ns_1\,(ns_2+nf_2)}{ns_2(ns_1+nf_1)+ns_1(ns_2+nf_2)}.$$

The crossover rate adaptation is similar to the method used in SaDE, but the factor Cr_m is changed as a biased average of the successful values Cr_{suc} every 25 iterations.

$$Cr_m = \frac{1}{k}\sum_{k=1}^{k} W_k Cr_{suc}(k)\; W_k = \frac{\Delta_K}{\sum_{k=1}^{k}\Delta_K}$$

where the weight is calculated with a positive improvement Δ= f (x) – f (u) in the selection related to each successful crossover rate CRsuc(k).

1.5.14 Self-Adapting Parameter Setting in Differential Evolution (jDE)

jDE is another adaptive DE algorithm that is similar to the classic DE/rand/1/bin algorithm. jDE improves the population size throughout the optimization process based on the improved parameters and thus generates vectors that are more likely to survive. However, the mechanism of jDE involves adapting the parameters Fi and CRi associated with each individual. At the beginning of the process, the parameter values are Fi = 0.5 and CRi = 0.9 for each individual. However, Fi and CRm are updated from the effective records; thus, jDE produces new values within the probabilities $\tau_1 = \tau_2$, which are used to alter the control parameters. The updated values for Fi and CRi are then obtained using uniform distributions over [0.1, 1] and [0, 1], respectively. That is,

$$F_{i,g+1} = \begin{cases} 0.1 + 0.9 * random1, if\;\; random2 < \tau_1 \\ F_{i,g} \qquad\qquad\qquad\qquad otherwise \end{cases}$$

$$CR_{i,g+1} = \begin{cases} random3, if\; random4 < \tau_2 \\ CR_{i,g} \qquad\qquad\qquad otherwise \end{cases}$$

where $random_j$ = 1, 2, 3, 4 is the uniform random function ∈[0, 1]. The updated parameters are implemented in the mutation and crossover processes to produce new, consistent vectors. This mechanism updates the prior parameter with a new one only if the new vectors pass the selection phase. However, jDE yields improved results with the classic DE/rand/1/ bin strategy.

1.5.15 Adaptive DE algorithm (ADE)

Hu and Yan proposed another adaptive DE algorithm. They modified the parameters F and Cr to each iteration using the current generation and the fitness [58]. They tried to find the optimal value for the parameters F and Cr to find a balance between reliability and efficiency. The mutation and crossover operations are calculated for each generation. Thus, for each parent x_i^g of generation g, the offspring x'^{g+1}_i is constructed as follows: calculate the Gth mutation $F(G)$ and crossover $CR(G)$ as

$$F(G) = F_0 * \left(\frac{\exp(I_G) - \exp(-I_G)}{40} + 1\right)$$

$$CR(G) = CR_0 * \left(\frac{\exp(I_G) - \exp(-I_G)}{40} + 1\right)$$

$$I_G = 3 - 6 * \left(\frac{G}{G_m}\right)$$

1.5.16 Modified DE (MDE)

MDE uses only one array, which is updated when a better solution is found. Therefore, continuously updating the one array improves the convergence speed, leading to fewer evaluation procedures than those associated with classical DE [59]. In MDE, and by applied distribution of Laplace "F" is arbitrarily adjusted [59]. The Laplace distribution is analogous to the (NP) Normal Distribution [60]. Moreover, the Laplace has a longer, skewed, allowing for inference so that it can control more efficiently, thus avoiding premature convergence. Experimental results demonstrate that modified DE with a Laplace distribution (MDE) offers enhanced performance compared with the classical DE approach [61].

1.5.17 Modified DE with p-best Crossover (MDE_pBX)

MDE_pBX involves F and Cr values that are produced using a Cauchy distribution using a position parameter, and then adapted relay on the power average of entirely F/Cr ratios producing effective offspring [62]. The mutation strategy used in this algorithm scheme (DE/current-to-best/1) can be expressed as follows:

$$V_{i,g} = X_{i,g} + F_i \left(X_{gr_{best},G} - X_{i,G} + X_{r_1^i,G} - X_{r_2^i,G}\right)$$

where $X_{gr_{best},G}$ is the finest of the q% vectors arbitrarily selected from the existing generation, whereas $X_{r_1^i,G}$ and $X_{r_2^i,G}$ are two distinctive vectors chosen randomly from the current population and are not equal to $X_{gr_{best},G}$ or the target.

In the p-best crossover process, for each different randomly vector chosen from the p best-ranking vectors in the present population[63] . Then, a standard crossover is executed as per (5) between the vector and the arbitrarily chosen one from p-top vector to produce the trial vector with identical index. The variable p is linearly make smaller with following generations as follows:

$$p = ceil\left[\frac{Np}{2}.(1-\frac{G-1}{G_{max}})\right]$$

where G is the present generation value, G_{max} is the most extreme number of generations and Np is the population number. The parameter adaption mechanism F_i is independently calculated as

$$Fi = \text{Cauchy Distribution } (F_m, 0.1)$$

$$F_m = w_f.fm + (1-w_f).mean_{Pow}(F_{success}) \text{ where } F_m \text{ initialized with 0.5}$$

$$w_f = 0.8+0.2* \text{ rand}(0,1)$$

$$mean_{Pow}(F_{success}) = \sum_{x \in F_{success}} (x^n / (|F_{success}|)^{\frac{1}{2}})$$

where n=1.5 and $|F_{success}|$ is the set of cardinalities.

The crossover probability adaptation Cr of each individual vector is independently created as

$$Cri = \text{Gaussian Distribution } (Cr_m, 0.1)$$

$$Cr_m = w_{Cr}.Cr_m + (1-w_{Cr}).mean_{Pow}(Cr_{success}) w_{cr} = 0.8+0.2* \text{ rand}(0,1)$$

$$mean_{Pow}(Cr_{success}) = \sum_{x \in Cr_{success}} (x^n / (|Cr_{success}|)^{\frac{1}{2}})$$

where n=1.5 and $|Cr_{success}|$ is the set of cardinality

1.5.18 DE with Self-Adaptive Mutation and Crossover (DESAMC)

DE with self-adaptive mutation and crossover (DESAMC) is a new version of DE [64, 65]. In this approach, F is adapted using an affection index (Af_i), calculated using information about fitness. A minor Af_i shows Which the each on is far away from the best global vector (best solution); consequently, a robust global exploration is essential. The formula of adaptation is as follows:

$$F_i(g) = \frac{1}{1 + \tanh(2Af_i(g))}$$

where tanh indicates the hyperbolic tangent function

$$\tanh(z) = \frac{\exp(2z) - 1}{\exp(2z) + 1}$$

where the crossover is $CR(t) = CR^- + (1 - \frac{t}{t_{max}})^{\frac{t}{t_{max}}} (CR^+ - CR^-)$

where t is the present generation, t_{max} is the greatest number of generations and CR^+ and CR^- are the maximum and minimum values of CR, respectively.

1.5.19 Adaptive Differential Evolution with Optional External Archive

Adaptive Differential Evolution with Optional External Archive (JADE) is an alternative to adapting the parameters at each generation toward progressive self-adaptation, based on the success rate [66]. Qin and Suganthan [45] and Zhang and Sanderson [66] proposed the new mutation strategy (DE/current-to-pbest/1). Furthermore, they used new adaptive parameters, μ_{CR} and μ_F

The crossover and selection operations are implemented as in the classic DE algorithm.The greedy strategy involves a new mutation strategy called DE/current-to-pbest/1 (without archive) and assists the baseline JADE:

$$V_{i,g} = X_{i,g} + F_i \left(V_{best,g} - V_{1,g}\right) + F_i \left(V_{2,g} - V_{3,g}\right)$$

$$V_{i,g} = X_{i,g} + F_i \left(V_{best,g} - V_{1,g}\right) + F_i \left(V_{2,g} - V'_{3,G}\right)$$

where $V_{best,g}$ is the best solution that is randomly chosen as one of the best individuals from the current population [49]. Similarly, $V_{1,g}$, $V_{2,g}$ and $V_{3,g}$ are randomly selected from the current population. However, $V'_{3,G}$ is also randomly chosen from the union between $X_{i,g}$ and $V_{1,g}$.

$$V'_{3,G} = \text{randomly } (V_{1,g} \cup X_{i,g})$$

JADE is also applied to the archiving process. Initially, the archive is unfilled and is added to the parent solutions that fail in the selection process [67]. The purpose of the archive is to avoid calculation overhead. Moreover, the archive has a limited size; thus, if the size of the archive grows beyond $R,$ then the shrink operation is performed to reduce its size so that it does not exceed (α, NP).The archive technique provides information for the directions to improve the diversity of the population. In addition, arbitrary F values can help expand population diversity [68].

1.5.20 Adaptation of μ_{CR} and μ_F

The adaptation technique used for JADE is applied to μ_{CR} and μ_F to produce the mutation rate Fi and the crossover rate CR_i related to each individual vector xi. JADE is implemented in each iteration i, and the crossover rate CRi of each individual xi is individually formed based on a normal random distribution = Normal Distribution (μ_{CR}, 0.1), where the mean μ_{CR} is initially 0.5 and the standard deviation is 0.1, i.e.,

$$CR_i = \text{Normal Distribution } (\mu_{CR}, 0.1).$$

Then, S_{CR} is calculated, which represents the set of all effective crossover rates CR_i. Furthermore, the parameter μ_{CR} is updated in each iteration; this information is saved, and random information is deleted from the archive file to keep its size R. μ_{CR} is calculated as follows:

$$\mu_{CR} = (1 - c).\mu_{CR} + c.mean(S_{CR})$$

Similarly, the mutation rate Fi is calculated using the Cauchy distribution (μ_F, 0.1), with the constraint that Fi =1. If Fi ≥ 1 or Fi ≤ 0 and μ_f is initialized as 0.5, then

$$F_i = \text{Cauchy Distribution } (\mu_f, 0.1)$$

where SF indicates the set of all effective mutation rates Fi. Then, μ_f is updated as follows: $\mu_f = (1 - c).\mu_f + c.mean_L(S_F)$

where $mean_L$ indicates the Lehmer mean calculated as follows:

$$mean_L(S_F) = \frac{\Sigma_{F \in S_F} F^2}{\Sigma_{F \in S_F} F}.$$

1.5.21 Differential Covariance Matrix Adaptation Evolutionary Algorithm (CMA-ES).

Saurav et al. proposed the Differential Covariance Matrix Adaptation Evolutionary Algorithm for real parameter optimization (CMA-ES) [69]. The goal of the covariance matrix adaptation is to estimate the reverse Hessian matrix, analogously to a quasi-Newton technique. Furthermore, to increase the utility of the DCMA-EA, the greedy selection method of DE is applied to improve individuals in the next generation [70]. CMA-ES uses a new differential perturbation structure, and the new population vector is shaped by the following equation:

$$X_{i,g} = m + \sigma.N(0,c) = m + \sigma.B.D.randn(dim)^T$$

where $randn(dim)^T$ is a group of random numbers taken from a normal distribution with zero mean and a standard deviation of 1 and has an element number equal to the dimensions of the function at hand. The parameter "m" and the evolution of "σ" determine the overall standard deviation.

By using sharing population, the new mutated vectors are produced to the target vectors as follows

$$V_{i,g} = m.(1-P) + P.X_{i,g} + F.\left(x_{r_1^i,g} - x_{r_2^i,g}\right) + (1-P).\sigma.B.d.randn(dim)^T$$

where $x_{r_1^i,g}$ and $x_{r_2^i,g}$ are two vectors randomly selected from the population, m is the average of the present population, B is an orthonormal of eigenvectors, and D is the square root of the commensurate none negative eigenvalues. P is a control value which

maintains the contribution of the average vector of the existing population and target ones as well. Both of the scale F and P are computed as follows:

$$F_i = 0.5 + 0.5.rand(0,1)$$

$$P = 0.5.(1 + \frac{iter}{iter - max})$$

CHAPTER 2: DIFFERENTIAL EVOLUTION WITH MULTIPLE STRATEGIES

In this approach, four different mutation strategies and one crossover operator are used within a single algorithm framework, as proposed by Elsayed et al. [71]. The main objective is to adapt a mutation strategy by choosing one from a pool of allowable schemes. In fact, although this algorithm involves different mutation strategies with dissimilar features, the authors believe that these different strategies cannot yield suitable performance. Therefore, the performance of the mutation strategy is dependent on the progression of the evolution, which is based on the success of the search operators.

Therefore, the feasibility status and the fitness value factors are used to measure the enhancement in the infeasibility. If the problem becomes increasingly feasible, the improvement index is calculated as follows:

$$VI_{i,t} = \frac{\left|V_{i,t}^{best} - V_{i,t-1}^{best}\right|}{avg.V_{i,t}} = I_{i,t}$$

where $V_{i,t}^{best}$ is the best individual at generation t and $avg.V_{i,t}$ is the average of the violation.

2.1 Hybrid DE Algorithms

Hybridization is another way to increase convergence for optimization. Hybridized approaches balance global and local search techniques. Hybridization is the method of joining the advantages of two or more algorithms to produce one algorithm that is anticipated to generate better offspring [72]. Each approach has its strengths and weaknesses. Thus, by combining different approaches, performance is improved [73]. Hybridization can be implemented at four stages of interaction [73]. The first is the individual stage for the search at examination level, which defines the performance of an individual in the population. The second is the population level, which appear as dynamic range of a population. The third is the exterior level, which delivers communication with other methods. The fourth is the meta data level, in which a superior metaheuristic contains its strategies [74].

Many attempts have been made to combine different algorithms to construct new hybrid algorithms. Genetic algorithms (GAs) and fuzzy philosophy are two recognized artificial intelligence methodologies [75, 76]. FUGE is constructed from a fuzzy model and a GA, which form a hybrid algorithm that consists of an iterative algorithm to update the offspring for job ordering for each VM (virtual machine). Then, the fuzzy algorithm obtains the fitness values for all offspring. This technique yields remarkable performance with cloud parameters such as those used in real-time communication.

Each optimization technique has specific operators and procedures; for example, the DE algorithm consists of mutation, crossover and selection. In the hybridized technique, some operators can cooperate between two algorithms to exploit the

complementary characteristics of different optimization strategies [77]. In fact, choosing a suitable combination of balanced algorithms is the key to achieving enhanced performance. Nevertheless, developing an effective hybrid algorithm is not easy because it requires proficiency in different areas of optimization. There are many types of problems for which a classic or modified differential evolution algorithm might fail to find a suitable solution [78]. Therefore, recently applied DE hybridization approaches have become widespread due to their ability to handle many real-world problems. Some of the benefits of DE hybridization have been previously discussed [40]. To enhance the performance of DE, such as the speed of convergence or the quality of DE, and to solve larger systems, DE must incorporate hybrid evolutionary methodologies [79]. In general, there are three types of hybridizations for evolutionary algorithms involving global optimization: hybridization with local search, hybridization with global optimization and hybridization involving both techniques [80]. In this section, we highlight and demonstrate several hybrid differential evolutionary algorithms reported in the literature.

2.2. Hybridization of DE with Other Evolution Algorithms

DE has been frequently hybridized with PSO because both algorithms implement simple difference processes to perturb the current population [81]. The variation between the current and the best individual is utilized both in the refresh population method of PSO and in the DE/current- to-best/1 mutation strategy.

The particle swarm optimization (PSO) method was offered by J. Kennedy and R.C. Eberhart [82, 83]. The technique shows perfect act compared with that of other evolutionary algorithms or metaheuristics. This approach mimics human cognition and

has been applied to the optimization problems. The goal is to apply a group of individuals called a swarm of particles [84]. The same notation used for DE is used for PSO; a vector is used as a solution for an optimization task t. At each loop t, a Particle alterations index, affected by its velocity $v_i(t)$ via the equation $x_i(t) = x_i(t-1) + v_i(t)$. However, two equations control the updating of the velocity $v_i(t)$.

$$v_i(t) = v_i(t-1) + \rho_1 * (p_i - x_i(t-1))$$

gbest represent the whole population; lbest describes the subpopulation encompassing the particle. The gbest is practical of best results. Let pg be the better results of the population; thus, social influence is mathematically expressed as $\rho_2 * (p_g - x_i(t-1))$. Therefore, updating the particles at each loop as follows:

$$v_i(t) = v_i(t-1) + \rho_1 * (p_i - x_i(t-1)) + \rho_2 * (p_g - x_i(t-1))$$

$$x_i(t) = x_i(t-1) + v_i(t)$$

where ρ1 and ρ2 are the control parameters.

PSO has several disadvantages, the most significant of which is its premature convergence. PSO consists of three components: previous velocities $v_i(t-1)$, present behavior

$\rho_1 * (p_i - x_i(t-1))$, and social behavior $\rho_2 * (p_g - x_i(t-1))$.

Because PSO is built on these three components, it will not operate if any of those components has any issue; for example, a vector consisting of a bad solution will retard the optimal solution. However, DE does not carry the initial two features of PSO. The

individual construct is based on a random walk algorithm in the search space, which then selects the optimal position index [85].

In PSO, the next position is based on the present optimal position pi and by the particle's velocity vi. In addition, the third feature of PSO could be inferred in DE as the RAND/BEST strategy. PSO refresh the velocity of a particle applying three expressions.

In the proposed strategy, the particle velocities are updated by carrying the subtract of the index vectors of any two dissimilar particles arbitrarily selected from the swarm. Das et al. proposed PSO-DV (particle swarm with differentially perturbed velocity) [86]. In the proposed scheme, particle velocities are perturbed by a new term containing the weighted difference of the position vectors of any two dissimilar particles randomly selected from the swarm. This differential velocity term mimics the DE mutation [87]. PSO-DV applies the DE differential operator to update the velocity of PSO. Two vectors are chosen randomly from the population. Then, unlike in PSO, a particle is moved to a new position only if the new position produces a better fitness value. In PSO-DV, for each particle i in the swarm, two other separate particles j and k (i ≠ j ≠ k) are chosen randomly. The difference between their locations is calculated as a difference vector:

$$v_{id}(t+1) = \omega * v_{id}(t) + \beta * \delta_d + C_2 \rho_2 * (p_{gd} - jx_{id}(t))$$

$$T_{ri} = x_i(t) + \beta * {}^{\delta}{}_{d} + C_2 \rho_2 * (p_{gd} - jx_{id}(t))$$

where CR is the crossover rate, ${}^{\delta}{}_{d}$ is the component of the subtract vector and β is a factor rate in the range [0, 1]. Hendtlass proposed the first combination of DE and PSO and called it SDEA, as the individuals comply swarm principles [55]. DE is used to

transfer the individuals to the promised region in random fashion. Xiaobing Yu et al. proposed an adaptive hybrid algorithm based on PSO and DE (HPSO-DE) with a composed populations among PSO and DE [88]. The strategy incorporates the advantages of the two algorithms and maintains population diversity. Therefore, HPSO-DE has the ability to move to local optima [89]. Zhang et al. offered DEPSO, which apples the similar standard of updating PSO individuals via DE [90]. DEPSO performs well with numerical integer problems but is not efficient for small feasible space problems. Mutations are maintained by a DE operator on p_i, with a trail vector $T_i = p_i$ for the dth dimension:

$$if\ rand(\quad)CR\ or\ d == k)\ then\ T_{id} = p_{gd} + \delta_{2,d}$$

where k is a random value within the domain [1, D], which include that the mutation has at least one dimension. CR is a crossover constant, and δ_2, is the case of N=2 for the general difference vector δ_N, which is defined as follows:

$$\delta_N = (\sum_{1}^{N} \Delta)/N$$

$$\Delta = P_A - P_B$$

where Δ is the difference vector and $P_A\ and\ P_B$ are chosen from the p-best set at random. Liu et al. offered a hybridization of PSO and DE in a pair of population scheme [91]. Three mutation strategies are borrowed from DE (DE/rand/1, DE/current_to_best/1DE/rand/2) are applied to refresh the former best solutions [92].

Trivedi et al. proposed a hybrid of DE and GA to resolve scheduling challenges [93]. GA operates on the binary element variables through the DE process to enhance the related

power-related variables [94]. The advantage of a GA lies in its ability to discover a decent solution to a problem whenever the iterative approach is too expensive in time and the mathematical approach is unobtainable [95]. GA allows for the fast discovery of the solution. Although the genetic algorithm is not excessively complex, the parameters and implementation of the GA generally require a tremendous amount of tuning [96].

The advantage of DE is that, in general, it frequently shows better solutions than those yielded by GA and other evolutionary algorithms [97-99]. Furthermore, DE is easy to apply to a wide variety of problems regardless of noisy, multi-modal, multi-dimensional spaces, which typically make problems difficult optimize. Although DE consists of two important parameters, Cr and F, those parameters do not require the same amount of tuning as those associated with other evolutionary algorithms [100]. and Liao has proposed a hybridization of DE and a local search algorithm modeled after the harmony search (HS) algorithm to find the global optimum [101]. The main goal of this type of hybridization method is to advance the use of mixed discrete and real-valued-dimensional problems.

Boussaïd et al. proposed a hybridization of DE and Biogeography Based Optimization (BBO) to deliver solution through the optimal power distribution method in a Wireless Sensor Network(WSN)[102, 103].

Dulikravich et al. proposed a hybridized multi-objective, multi variable optimizer by combining non-dominated sorting differential evolution (NSDE) with the strength Pareto evolutionary algorithm (SPEA) and multi-objective particle swarm optimization (MOPSO) [104].

Haixiang Guo and others have proposed a form of DE enhanced among self-adaptive parameters that depend on simulated annealing algorithms in the collection of DE; the classic selection technique is a greedy equation [105]. The greedy rule is easily trapped in a local optimum. However, a new selection technique based on simulated annealing is used in this algorithm. The approach is expressed as follows:

$$X_{i,G+1} = U_{i,G} \ \ if\ f(U_{i,G}) \leq f(X_{i,G})$$

$$X_{i,G+1} = U_{i,G} \ \ if\ f(U_{i,G}) > f(X_{i,G})\ and \exp\left(-\frac{f(U_{i,G}) - f(X_{i,G})}{t_G}\right) > rand(0,1)$$

$$X_{i,G+1} = X_{i,G} \ \ if\ f(U_{i,G}) > f(X_{i,G})\ and \exp\left(-\frac{f(U_{i,G}) - f(X_{i,G})}{t_G}\right) \leq rand(0,1)$$

where t_G represents the G_{th} generation temperature. Pholdee and Bureerat offered a hybrid algorithm involving the trial vector method of DE called the Real-Coded Population-Based Incremental Learning (RCPBIL) algorithm [106]. The RPBIL can be extended to multi-objective optimization similarly to multi-objective PBIL using binary codes for which the population is serve as a likelihood vector for single-objective problems [107]. When addressing multi-objective problems, more probability vectors are utilized to maintain population variety. Likewise to the binary code of PBIL, the multi-objective style of the RPBIL uses numerous possibility matrix that appear for a real code population, where each probability matrix is called a tray [108].

Three-dimensional matrix which is represent a group of probability trays is a with dimensions n*n_l*n_T, that n_T is the number of trays required for each tray drive to be used to produce a real-code subpopulation, which has approximately N_P/N_T form results as its members.

An initial population is formed for the search procedure of multi-starting with early likelihood trays. An initial Pareto archive is gained, and non-dominated results are then designated to update the probability trays. Then, the centroid of the non-dominated solution set (R_G) is used to update a probability tray in the series, where the r_G of the set that has the lowest value of the first objective function is applied to update the first tray and so on.

The updating procedure for each tray can be improved by substituting X_{best} with r_G. Subsequently, a population yielding the updated trays is shaped. The Pareto archive is changed by substituting its members with non-dominated solutions saved from the mixture of the current population and the elements in the preceding archive. If the number of archive elements is larger than the constant archive size, the clustering method is initiated to eliminate non-dominated solutions from the archive. These steps are repeated until a stopping condition is fulfilled [109].

Neri et al. [110] proposed a compact DE hybridized with a memetic search to yield faster convergence [111]. The algorithm represents the population as a multi-dimensional Gaussian distribution and is called Disturbed Exploitation compact Differential Evolution (DEcDE) [112]. The DEcDE algorithm utilizes an evolutionary framework based on DE logic assisted by a shallow depth for processing the local search algorithm [112].

The output of the algorithm was introduced to create an MC model to gain high efficiency on a diverse set of problems, regardless of its limits, in terms of complexity

and memory usage. At the start of the DEcDE algorithm, an $a(2.n)$ probability vector (PV) is produced.

$$PV^t = [\mu^t, \sigma^t]$$

where μ and σ are, respectively, the mean and standard deviation values for each design variable from a Gaussian probability distribution function (PDF) truncated within the interval [-1, 1].

$$PDF = [\mu[i], \sigma[i]]$$

Zhi-hui Zhan and Jun Zhang proposed a differential evolution (DE) algorithm with a random walk (DE-RW) [113]. DE-RW is analogous to the classic DE algorithm, with a minor alteration in the crossover procedure that mixes the individual vector and the mutant vector to perform a random walk, forming the target vector as follows:

$$v_{id} \begin{cases} \text{if } rand(0,1) < CR \text{ or } d = k \\ rand(L_d, H_d), \quad \text{else if } rand(0,1) < RW \\ x_{id}, \qquad \text{otherwise} \end{cases}$$

where L_d and H_d are the low and high search restrictions of the d_{ht} dimension and the parameter RW is used to control the effect of the random walk. The parameter RW is controlled as follows:

$$Rw = 0.1 - 0.099 \times g/G$$

where g and G are the current generation number and the maximum number of generations, respectively. A few some remarkable DE algorithms are shortened in Table2.

Table 2.1. Summary of different DE algorithms with verity of approaches

Algorithm	Strategy	Note
Multi PopulationDEalgo-rithm (MPDE)[114]	DE/best/1	MPDE created subpopulation in random manner form main population and, then the migration of the best vector from subpopulations to main population
Adaptive DE[115]	six DE strategies and one strategy is randomly selected by a roulette wheel	Adaptively selects a trial vector generation, scale factor "F" is 0.8 and it is constant for all strategies also the crossover rate is constant = 0.5
Self-Adapting Parameter Setting in Differential Evolution (jDE)	DE/rand/1/ bin	jDE enhanced the population size based on the developed the DE parameters
Self-adaptive Mutation DE (SaMDE)	DE/rand/1,DE/best/1,DE/best/2andDE/ current-to-rand/1	The strategy is chosen by a roulette wheel strategy. The scale factor is dynamic and chosen form range [0.7; 1.0) after each generation.
Modified DE(MDE)with pbest crossover(MDE-pBX)[116]	DE/current-to-best/1, DE/current-to-gr_best/1 [gr indicate for group]	F and Cr directed by the information of their effective values that were capable to produce improved offspring
Modified DE algorithm (MDE)	DE/rand/1 , DE/ best/1	One of the two strategies, is chosen based on a probability.
DE with Self-Adaptive Mutation and Crossover (DESAMC)	Classic DE Strategy	Working to self-adaptive the parameters values
Differential Covariance Matrix Adaptation Evolutionary Algorithm (CMA-ES)	new population vector is created using DE/rand/1/ bin	Parameters are chosen randomize
Differential Evolution with Multiple Strategies	DE/best/1/bin,rand/1/bin, DE/best/1/exp ,and DE/rand/1/exp	Parameters are chosen randomize
DE-PSO	Classic DE strategy + The two basic equations which govern the working of PSO	"DE-PSO" Hybrid differential evolution - Particle Swarm Optimization. The inclusion of PSO phase creates a perturbation in the population.
Hybrid of DE and GA (hGADE)[117]	hybridized GA with only 2 classical DE variants	randomly generated binary unit commitment matrices while the RPM of all the individuals in the initial population are generated
hybridization of DE and Biogeography Based Optimization (BBO)[118]	classical DE/rand/1/ bin + classic BBO	The main operator of DE/BBO is the hybrid migration operator,

CHAPTER 3: RESEARCH PLAN

3.1 Introduction

The DE algorithm has been applied in several applications such as scheduling, image processing, multi-modal methods, non-convex methods, and among many others [119-121]. The traditional performance of DE is based on the chosen strategy and the control parameters[122, 123]. This strategy consists of mutation, crossover, and selection, and there are three control parameters: the number of populations “NP”, the mutation factor “F” (sometimes called the scaling factor), and the crossover rate “Cr”[124-126]. Indeed, the performance of DE relies on the values of the population size “NP”, the mutation factor “F”, and the crossover rate” Cr”[127, 128]. Many studies have been conducted in classic DE, such as using mutation with perturbation, mutation with selection pressure, and a neighborhood mutation operator[129].

The second phase is the crossover and there are two different crossover techniques, either binomially or exponentially, produced different quality of results[40, 130]. The major aspect of the crossover is to determine the element of the trail vector that will be inherited from the target vector[131]. Additionally, the performance of DE relies on strategies and the right control parameter values [132-135]. Extensive researches have been conducted to determine what the best control parameter values. There are two approaches for setting these control parameters: predefined (also called deterministic) and adaptive approaches. In fact, in the deterministic technique there are some recommended

values for these parameters for which it is not required to obtain any feedback. However, with adaptive technique the parameters values assigned and adjusted dynamically through the processing according to pre-defined rules. Unfortunately, the adaptive and self-adaptive techniques that are often time-consuming for the evolution for each parameter value because of their high complexity[136]. In fact, adaptive and self-adaptive are very effective for small dimensional problems, however they are produce poor results when the dimensions are increased[137]. Therefore, researchers have focused on finding suitable and efficient strategies to speed up the convergence rate[138, 139].

In this work we introduce a new proposes a Multi-Layer Strategies Differential Evolution (MLSDE) approach, which uses different mutation strategies in order to reach a fast convergence rate and avoid premature convergence due to the loss of diversity in the population. Multilayer techniques were applied since there is no single method that has proven fit for every problem. Some strategies may work perfectly with some problems, while other strategies perform poorly with other problems. ndeed the MLSDE works to improve the diversification of offspring by using different strategies in a multiple-layered approach. This approach spread out the population so that the sampled vectors can easily generate improved offspring. One of the advantages of this technique is its ability to reach a very quick rate of convergence to find the optimal solution with a minimum number of iterations.

3.2 Multi-Layer Strategies Differential Evolution

The multi-layer strategies differential evolution (MLSDE) approach operates in the same manner as classic differential evolution with an initialization population, mutation, crossover, and then the selection operation. However, MLSDE consists of a group of mutations, crossovers, and selections that are performed in sequence.

In MLSDE, the first step is to initialize the main matrix with random population within constraints of upper and lower bound values. Then, the different vectors are chosen as the core of the mutation operations. These vectors differ in the way in which they present in domain space because of their different composition. Therefore, to obtain the diversity of the domain space, six vectors are chosen V1, V2, V3, V4, V_{best}, and V_{Hill} from the population. Two approaches are used to construct the best vectors. The first approach, V_{best} uses the objective functions to find best vector in the main matrix, whereby each row represents an independent vector. The second approach, uses Hill Climb method to construct vector V_{Hill}. The best vectors would often leads to a fast convergence and performs well when solving for unimodal problems. The combination of V_{best}, and V_{Hill} helps to balance between exploration and exploitation.

Following the mutation stage the, the crossover between vectors occurs to produce improved vectors [136]. Crossover results in high diversity in populations by applying the crossover equations (11),(12), and (13).The crossover probability, Cr ∈ [0,1], is pre-defined value that controls the fraction of parameter values that are copied from the mutant. To control which source contributes a given parameter, uniform crossover. Compares Cr to the output of a uniform random number generator ,randj(0,1). If the random number is less than or equal to Cr, the trial parameter is inherited from the mutant, Vi, j otherwise, the parameter is copied from the vector, Xi, j . In addition, the trial parameter with randomly chosen index, jrand, is taken from the mutant to ensure that the trial vector does not duplicate Xi, j Because of this additional demand, Cr only approximates the true probability that a trial parameter will be inherited from the mutant[140].

$$U_{1(i,j)} = \begin{cases} V_{y1(i,j)} & \text{if f rand1} \le Cr \end{cases}$$

$$U_{2(i,j)} = \begin{cases} V_{y2(i,j)} & \text{if f rand2} \le Cr \end{cases}$$

$$U_{3(i,j)} = \begin{cases} V_{y3(i,j)} & \text{if f rand3} \le Cr \end{cases}$$

Once chosen, the different vectors are used in the below equations to calculate new vectors Vy1 , Vy2, and Vy3. This stage is referred to as the MLSDE mutation.

Next stage is selection stage. The trail vectors produced from equation (11),(12) and (13) are compared with target vectors. If the trial vector, U1(i,j), U2(i,j), and U3(i,j) have an equal or lower objective function value than that of its target vector, X1(i,j), X2(i,j) , and U3(i,j) they replace the target vector in the next generation; otherwise, the target retains its place in the population. The flowchart of MLSDE algorithm is prsented in Figure.2.

$$X_{1,\,i+1} = \begin{cases} U_{1,i} & \text{if } f(U_i) \le f(X_i) \\ \end{cases}$$

$$X_{2,\,i+1} = \begin{cases} U_{2,i} & \text{if } f(U_i) \le f(X_i) \\ \end{cases}$$

$$X_{3,\,i+1} = \begin{cases} U_{3,i} & \text{if } f(U_i) \le f(X_i) \\ \end{cases}$$

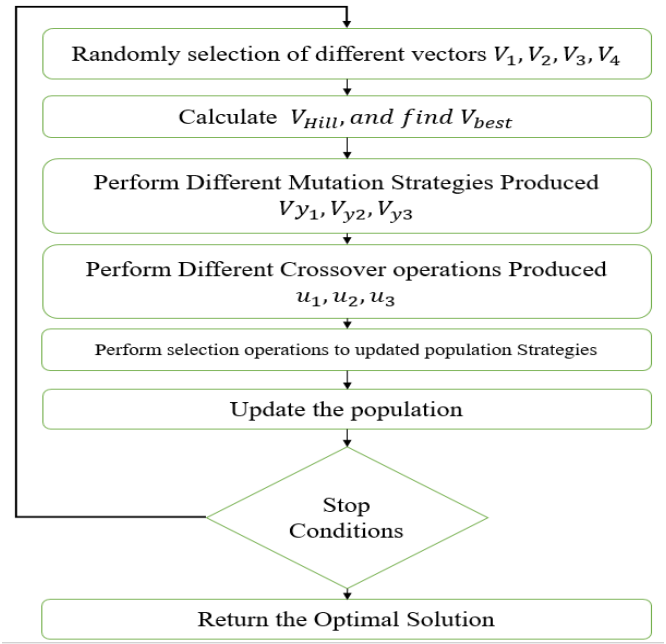


Figure 3.1. The flowchart of Proposed MLSDE

CHAPTER 4: IMPLEMENTATION AND RESULTS

4.1 Implementation and Test Plan

We have conducted experimental tests on the optimization benchmark suite four typical minimization problems introduced in the CEC 2013 benchmark functions suite experiments. Therefore, the growing research area is divided into adaptive, self-adaptive, and hybridization strategies. Thus, this study may provide a roadmap through which developers may gain a full understanding of this field. To evaluate the reliability and robustness of the different DE algorithms, we introduce a general framework that includes the control parameters for evaluating the efficiency of the different algorithms. In addition, the proposed MLSDE algorithm is examined on the classical benchmark functions provided by the CEC2015 Special Session.

4.2 Results

The well-studied domain of function optimization was used to test the performance of the proposed MLSDE algorithm , which was evaluated based on the classical benchmark functions provided by the CEC2015 Special Session [141]. The algorithms was used for the comparison of different algorithms include JADE, JDE, and SADE [141, 142]. In this section, MLSDE is employed to minimize a set of 16 scalable

benchmark functions with dimensions of D=30 and D=100. The parameters of MLSDE were fixed at F=0.5 and Cr=0.8, which represent better parameter values as reported in literature [29, 143]. The optimal values for all of these functions were equal to 0. The functions F1-F5 were unimodal functions and F5 exhibited multiple minima in high-dimensional case. Functions F6-F12 were multimodal functions and F13 is an expanding multimodal function (quartic function). The number of demission D was set for 30 and 100 at all 16 test functions. Table 3 Experimental Results of MLSDE, JADE, JDE and SADE for 50 independent runs of 30 Variables.

Table 4.1. Mean experimental results for 30 Variables over 50 runs

Func	MLSDE	JADE	JDE	SADE
F1	1.28E-82	2.69E-56	1.46E-28	3.42E-37
F2	3.68E-28	3.18E-25	9.02E-24	3.51E-25
F3	2.37E-74	6.11E-81	1.16E-13	1.54E-14
F4	7.53E-26	5.29E-14	2.44E-14	6.39E-27
F5	2.09E-04	1.59E-01	1.04E-03	7.98E-02
F6	0.0E+00	0.00E+00	0.00E+00	0.00E+00
F7	5.07E-06	6.14E-04	3.35E-03	2.06E-03
F8	0.0E+00	0.00E+00	0.00E+00	0.00E+00
F9	0.0E+00	0.00E+00	0.00E+00	0.00E+00
F10	4.38E-12	4.14E-15	8.26E-15	4.04E-15
F11	0.00E+00	0.00E+00	0.00E+00	0.00E+00
F12	0.00E+00	1.57E-32	5.99E-30	1.57E-32
F13	3.02E-30	2.17E-32	1.80E-27	1.35E-32
F14	1.06E-11	1.68E-09	7.31E-01	1.25E+02
F15	2.08E-02	2.00E-01	1.98E-01	1.56E-01
F16	2.14E-08	2.78E-05	6.08E-10	2.94E-06

The performance of MLSDE was further compared with three other state-of-the-art DEs, JADE [17], JDE [15], and SADE [18]. The experimental results aver-aged over 50 runs are listed in Tables 1 for D=30 and Table 2 for D=100. Ac-cording to the graphs in Figs. 6.3 and 6.4, MLSDE resolved the optimization problem and showed superior performance compared to other algorithms JADE, JDE, and SADE. These compared algorithms were able to find a near-global optimum on most of the benchmark functions, because they all utilized different strategies to improve the algorithms' robustness.

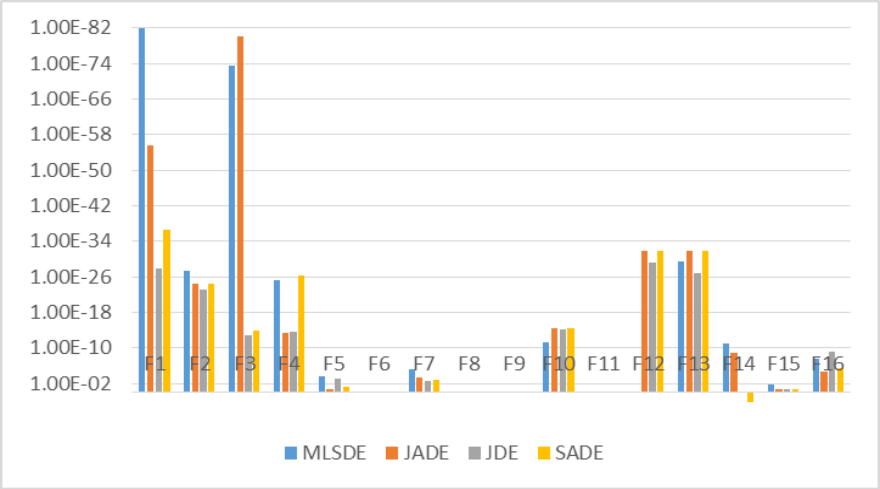


Figure 4.1 Comparing between MLSDE, JADE, JDE, and SADE for D=30

However, JADE, JDE, and SADE sometimes became stuck at local optima on some functions such as F2, while our proposed MLSDE reached a near-global optimum in every run in both 30 and 100 dimensions. However, for D=30 the JADE performed best for F3 and SADE performed best for functions F4, F10, and F13. JDE achieved better for F16.

Generally, the proposed MLSDE algorithm accomplished the best performance on 11 functions (F1, F2, F5, F6, F7, F8, F9, F11,F12, F14, and F15) out of the 16 benchmark functions with D=30 .In case of D=100, the results also showed better and the MLSDE able to find the optima for 10 functions (F1, F2, F3, F4, F5, F6, F7, F9, F12, F14).

Even for functions where MLSDE was not able to achieve a better solution than the other algorithms, it has the capability to achieve near-global optimum with competitive solution accuracy. For D=100, the JADE performed better for F10, F13, F15, F16 and SADE performed better with function F11. However, the MLSDE achieved near-global optimum with these functions as well.

The experimental results averaged 50 independent runs are listed in Table 1 for D=30 and Table 4 for D=100. According to the graphs presented in Figs. 1 and 2, MLSDE resolved the optimization problem for some functions and showed superior performance compared to JADE, JDE, and SADE

Table 4.2 Mean experimental results for 100 Variables over 50 runs

Func	MLSDE	JADE	JDE	SADE
F1	4.02E-70	5.13E-62	2.09E-20	1.09E-27
F2	1.29E-27	5.19E-16	1.82E-12	1.09E-15
F3	2.14E-58	6.85E-03	7.47E+03	4.96E+00
F4	9.01E-02	1.62E-01	1.60E+00	1.90E-01
F5	1.03E-02	4.96E+01	9.20E+01	8.49E+01
F6	0.0E+00	0.00E+00	0.00E+00	0.00E+00
F7	1.81E-03	2.04E-03	2.08E-02	6.85E-03
F8	5.07E+04	3.94E+03	2.81E-08	1.89E+01
F9	1.19E+00	1.03E+02	6.01E+00	1.05E+02
F10	6.01E-12	7.69E-15	1.73E-11	1.05E-14
F11	1.02E-02	8.87E-04	0.00E+00	2.96E-04
F12	7.12E-43	4.71E-33	4.47E-21	6.75E-30
F13	1.72E-17	1.35E-32	1.91E-17	5.56E-27
F14	8.02E+04	1.51E+05	2.07E+05	1.69E+05
F15	4.29E-01	3.28E-01	3.80E-01	3.60E-01
F16	6.89E-08	1.12E-11	4.78E-03	5.78E-03

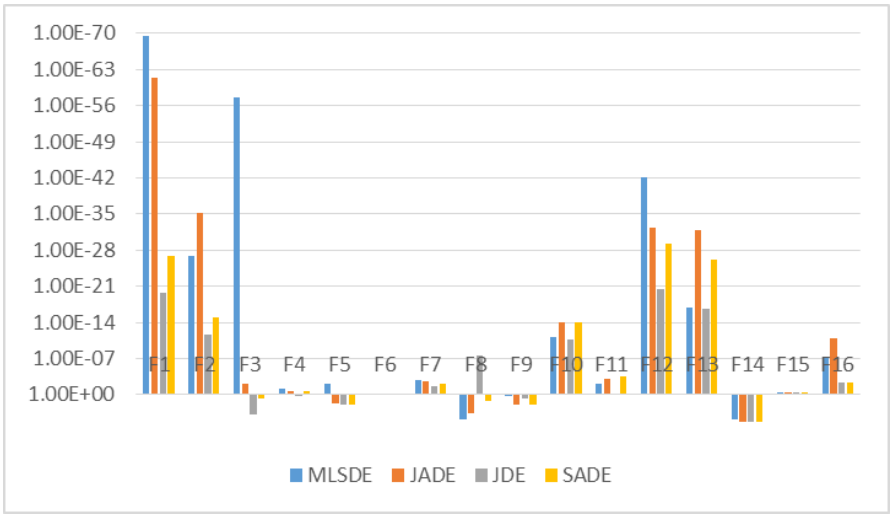


Figure.4.2. Experimental Results for 100 Variables

CHAPTER 5: APPLICATION

In denote image processing which is indicated by two-dimensional functions of the formula f(x, y). The value of function f(x,y) (called amplitude) is always a positive quantity that is calculated by the source of the image. When an image is produced from a sensor its intensity values are proportional to energy emitted by this sensor. As a import that f(x,y) essential be finite and nonzero.

However, the function f(x, y) is consisted of two components: (1) the quantity of source illumination, and (2) the reflectance which meaning the quantity of illumination reflected by the object. Therefore, these are the illumination and reflectance are denoted by i(x. y) and r(x, y), respectively. The product of f (x, y) as following

$$f(x,y) = i(x,y)r(x,y)$$

$$where$$

$$0 < i(x,y) < \infty$$

$$0 < r(x,y) < 1$$

5.1 Image Sampling and Quantization

There are many methods to obtain images, but the goal is to produce digital images from sensors. The production from sensors is a continuous voltage whose amplitude and spatial are linked to the object phenomenon that being sensed.

In order to construct a digital image, we essential to change the continuous data into digital form. This includes two procedures: sampling and quantization. The essential idea of sampling and quantization that a continuous image f which required to change to

digital image which continuous the x- and y-coordinates, and amplitude. However, the sampling is the first step in this to converting process which is digitizing the coordinate values. However, the quantization process that is digitizing the amplitude values. The Fig. 7.1(b) illustrate one-dimensional function of continuous image that is a plot of amplitude (intensity level) values. This continuous image along the line part AB in Fig 7.1(a).

Since there is noise in the image that cause the random variations. Thus, similarly spaced samples along line AB is taken in order sample this function. The spatial position of each apiece sample is specified by a vertical value in the bottom part as show in the figure 7.2.

The white squares in figure 7.2 representing the set of discrete positions that provides the sampled function. Though, these values are in vertically a continuous range and it required to convert the value of the intensity as well. The conversion an intensity values to digitalized called quantized. The Figure. 7.2 displays the intensity scale in discrete intervals, extending from black to white. In the quantization process is assigning values from the right-side scale to each sample. In fact, the digitizing is made reliant on the vertical proximity of a sample. As result, as the number of discrete levels applied, the accuracy attained in quantization. Therefore, the quantization is extremely dependent on the noise contented of the sampled signal.

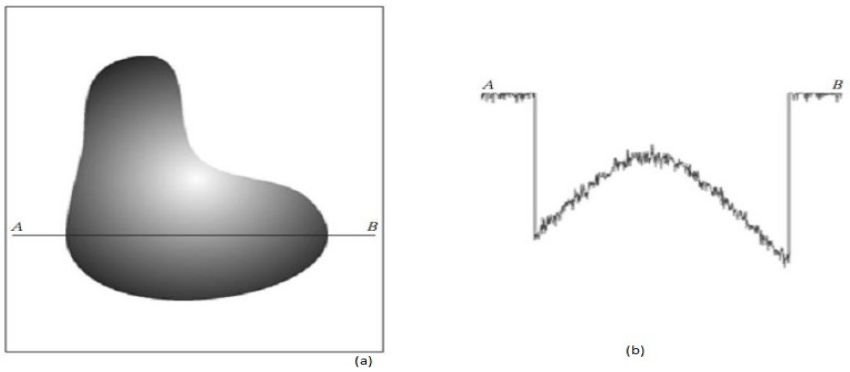


Figure 5.1 Generating a digital image (a) Continuous image (b) A scan line from A to Bin the continuous image, used to illustrate the

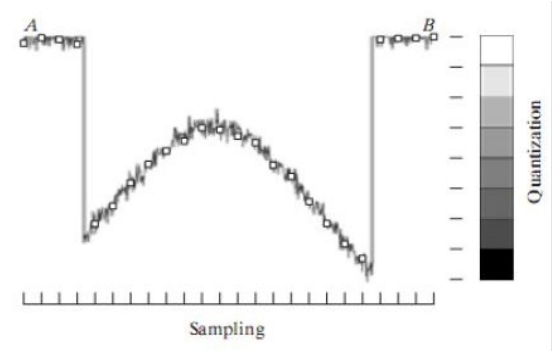


Figure 5.2 Generating a digital image Sampling and quantization.

Sampling in the manner just described assumes that we have a continuous image in both coordinate directions as well as in amplitude. Practically, the sampling technique is defined by the sensor arrangement that applied to construct the image. Moreover, spatial sampling is achieved by choosing the number of individual mechanical increases at which how the sensor was activated to gather data. Obviously, the quality of a digital image is specified by the number of samples and discrete intensity levels applied.

5.2 Representing Digital Images

The image in the continuous form represent as function f(s,t). However, the main goal to convert this f(s,t) into a digital image form by applying both sampling and quantization. Then in sampling phase the continuous image turns to 2-D array, as discrete coordinates f(x, y) with size of M rows and N columns where x = 0, 1, 2, ... , M - 1 and y= 0, 1, 2, ... , N-1. The coordinate is indicated to the number of samples. For example, the sample (0, 1) is indicate to the second sample along the first row where sample (0,0) is origin sample. In addition, the actual value of the image at any coordinates (x, y) is represented f(x, y), where x and y are integers.

However, the image so complex and it has too much detailed and thus it is difficult to interpret from plots. However, gray-scale image sets can be expressed as triplets of the form (x, y, z), where x and y are coordinates and z is the value of f at that coordinates (x, y).

Therefore, the intensity of each point is proportional to the value of f at that point. For example, if the intensity is normalized to the in boundary [0, l], so apiece of point in the image may have the value 0, 0.5, or 1. Those values converts to black, gray, or white, respectively. The third representation is displaying the f(x, y) as an array if the size of f is 600 X 600 elements which is equal 360,000 number.

$$f(x, y) = \begin{bmatrix} f(0,0) & f(0,1) & \cdots & f(0, N-1) \\ f(1,0) & f(1,1) & \cdots & f(1, N-1) \\ \vdots & \vdots & & \vdots \\ f(M-1,0) & f(M-1,1) & \cdots & f(M-1, N-1) \end{bmatrix}$$

$$\mathbf{A} = \begin{bmatrix} a_{0,0} & a_{0,1} & \cdots & a_{0,N-1} \\ a_{1,0} & a_{1,1} & \cdots & a_{1,N-1} \\ \vdots & \vdots & & \vdots \\ a_{M-1,0} & a_{M-1,1} & \cdots & a_{M-1,N-1} \end{bmatrix}$$

Figure 5.3 Intensity of each point(pixels) speared in Matrix format

This digitization procedure needs to decide what the number of discrete intensity levels L. However, because of storage and quantizing hardware limitation considerations, the number of intensity levels naturally is an integer power of

$L=2^k$ (intensity level) where k number of bits

The number of bits 'b' essential to store a digitized image is ; b= M*N*K

For example, if intensity level of k-bit image such as 256 possible discreet intensity values could be 8-bit image. However, the storage requirement for 8-bit image is 1024*1024.

5.3 Image quantization

The objective of color quantization is to constitute the many colors in the image with a decrease number of dissimilar colors and with minimum distortion. Original color images consist of thousands of colors up to 16,777,216 colors. However, as many colors

using an image that can lead to an improved output image .Though, too many colors can lead to image-processing problems. In fact, during image-processing such as object detection and object extraction that the number of colors consider crucial feature. In case many colors represent single object that lead to real problem. Thus, in image processing techniques conduct reduce number of colors as preprocessing step.

However, the color quantization contains of pixel-mapping phases by design palette. In the design palette phase is the collection of selected colors that use demonstrate the image with minimum distortion. In fact, the pixel-mapping phase is the assign each pixel in the image to one of the colors in the designed palette. Color quantization approaches apply a clustering procedure to design the palette and then map each pixel with the designed palette Consequently, the level of distortion is specified by the clustering algorithm that is applied for design palette.

Color image quantization is the procedure of decreasing the number of colors existing in a digital color image. The color quantization is applied to decrease the colors number of a digital image with a minimum visual deformation. It used to regenerate images on visuals hardware that has restricted number of simultaneous colors (e.g. frame memory displays with 4 or 8-bit colormaps). The color quantization reduces space necessities for storing of image data and decreases broadcast band width necessities in multimedia applications. Similarly, quantization defined as the procedure of mapping a continuous variable to a discrete set of values. Color quantization usually denotes to the issue of selecting k colors from a color space to constitute n ($k < n$) colors from the same color space and its target to reduce the total error. Color image quantization can be

representing in formal way as follows [27]: Assumed a set of NS colors where S' ⊂ RNd and Nd is the dimension of the data space. Color image quantization can be representing in formal way as follows: fq : S'→ S″ where S″ is a set of Ns″ colors such that S″ ⊂ S′ and $N_{s''} < N_{s'}$. The objective is to minimize the quantization error resulting from replacing a color C ∈ S′ with its quantized value fq (C) ∈ S". Color image quantization is a significant issue in the image processing fields.

Color image quantization contains of two main steps. The first step is forming a colormap which also called a palette that form a small set of colors in the range of 8-256. This palette is selected from the 224 potential mixtures of red, green and blue.which abbreviated as “RGB”. The second step that it is mapping each color pixel to one of the appropriate colors in the palette.

There are numerous techniques for color quantization. The class splitting algorithms which split the color space into separate regions, by repeated dividing up the space. Then each region a color is selected to indicate to the region in the color palette. The median-cut algorithm (MCA) and the variance-based algorithm (VBA) are two famous algorithms of this type. In fact, splitting algorithms have very good speed performance. However, the drawback of this type that it is hard to reach global optima, since a choice for splitting at one phase cannot be undone at a further phase. The color quantization classified as NP-complete. Therefore, is not feasible to search for the global optimal solution since this will need an excessive amount of time.

5.3.1 Related work

Numerous color image quantization techniques have been proposed. Most of these techniques applied clustering method. The clustering defines as the method of finding collections similar or related of objects to one another and those collection are different from unrelated to the objects. Furthermore, the problem of m-dimensional clustering to decrease the maximum inter cluster error or distance. It can be stated as discovery a divider of n points in m-dimensional Euclidean space to k separated clusters such as B1, B2, . . . , Bk where maximum (M1, M2, . . . , Mk), , thus Mi is the max distance between two centroids in cluster Bi, is reduced . In fact, a divider with small maximum inter cluster distance contains of small clusters where centroid for every cluster are near to each other in order to be an effective group. Intuitively, this is an optimization clustering which is corresponding to the minimal maximum quantization measure when m = 1. Because all points in cluster Bi are now spread along a line , and furthermore , Mi is the distance of the line segment (i.e., the length between the datapoint at throughout of the line and the datapoint at the opposite end). Centroid of the line segment select around middle of the line segment. Obviously, reducing the length of this line is corresponding to minimizing the length between the centroid and the endpoints. In fact, the main classification of image quantization techniques are post-clustering and pre-clustering.

5.4 K-Means Clustering Algorithm

The KM algorithm is one of the most extensively clustering techniques has been applied [48]. For example , given a dataset X=$\{x_1, x_2, \ldots . x_N\} \in \mathbb{R}^D$ the goal of KM is to

divider X into K comprehensive and totally unrelated clusters S= $\{s_1, s_2, \ldots, s_N\}$, $\bigcup_{k=1}^{K} S_k$= X , $S_i \cap S_j = \Theta$ for $1 \leq i \neq j \leq K$ by reducing the sum of squared error (SSE)

$$SSE = \sum_{k=1}^{k} \sum_{x_i \in S_k} \| x_i - c_k \|_2^2$$

Where $\|\cdot\|$ denotes to Euclidean (L_2) norm and c_k is the center of cluster S_k found it as the mean of the data of cluster.

In fact, K is random centers which selected from the data [51]. Every point in this data then labeled to the adjacent center, and respectively new center is recalculated as the mean of cluster. Repeating those steps until a termination condition is satisfied. The pseudocode for this algorithm is given in Algo. (1) Notes: 1. m[i] indicate the membership of point x_i, that mean the index of the cluster center that is nearest to x_i

The main drawbacks of KM are that it frequently ends at a local minimum and that its outcome is influenced by the preliminary selected of the cluster centers. Furthermore, pixel mapping stage is wasteful, because of a full search of the palette is essential to find the adjacent color. However, the pre-clustering techniques frequently operate and store the palette in a superior data structure such as binary trees which lets moving to nearest neighbor faster and the search during the mapping stage more feasible.

The proposed Color Quantization using MLSDE (CQ-MLSDE) is presented, which is post-clustering quantization technique. It produces clustering of the color map.In the MLSDE context, a single individual represents a colormap (i.e. an individual contains K cluster centroids indicated as RGB color colormap). Each individual is vector Xi built as Xi = (mi,1,…,mi,k), where mi,k denotes the kth cluster centroid vector of the ith

individual. Therefore, the population consists of several candidates' colormaps. The quality of each individual is evaluated using the MSE (Eq. 8) as follows:

$$f(Xi)= MSE(Xi) \quad (9)$$

1. Input : X= {x1,x2,.xn} ϵ ℝD (N×D input dataset)
2. Output: C= {c1,c2,.cn} ϵ ℝD (K cluster centroid)
3. Choose arbitrary subset of X as primary set of cluster centroid
4. While termination condition is not satisfied do
5. For (i=1 ; i<= N; i=i+1) do (For loop#1)
6. Assign Xi to close cluster
7. m[i]= $\| x_i - c_i \|^2$
8. End (For loop#1)
9. Determine the new centroid of cluster
10. For(k=1 ; k<= K; k=k+1) (For loop#2)
11. Sk cluster consist of set points xi which are closer to centroid Ck
12. Sk = { Xi | m[i]= k}
13. Calculate Ck as new center for Sk
14. Ck = 1/SK $\sum_{xiesk} xi$
15. End (For loop#2)
16. End While

In fact, this technique begins by initializing each individual arbitrarily from the color image to contain K centroids (i.e. color plate). The set of K color plate indicates the colormap. The K-means clustering is then applied to each Induvial in a probability

manner, pkmeans. The Kmeans is applied to improve the selected colors and to decrease the search space. Then every pixel is allocated to the cluster with the nearest centroid. The fitness function of each individual is determined by applying Eq. 9.

Afterwards, the population is reordered and updated, and this process is repeated till a stopping criterion is matching. The color map of the global best individual after exceeding max iterations is selected as the optimal result.

The Image quantization using CQ-MLSDE is summarized below:

1) Initialize Population Matrix with each individual by arbitrarily selecting K color centroid $C_{i,k}$ where i=0 (each row considers K cluster possible solution)
2) While (stop Criteria not satisfied) do
3) Calculate MSE (all clusters)
4) Assign Zp to proper $C_{i,k}$ { where min $d^2(z_p - m_{i,k})$}
5) Find V_best = Find_best (Population Matrix)
6) Choose randomly V1, V2, V3 , Vx
7) Calculate K-means with small number of iterations with probability rate pkmens
8) Calculate $d^2(z_p - m_{i,k})$ for all clusters $C_{i,k}$
9) Assign Zp to proper $C_{i,k}$ { where min $d^2(z_p - m_{i,k})$}
10) Calculate the fitness f(x)
11) Update the best solution yˆ(t)
12) Update the centroids using MLSDE

Although the parameters s, K and t_{max} are fixed, the best practice is that s, K and $t_{max} << N_p$. This MLSDE and image quantization technique have been applied to four

frequently used color images; Lenna (shown in Figure 1(a)), peppers, jet, and mandrill. The image size is 512 × 512 pixels, and they are quantized to 16, 32 and 64 colors. To demonstrate the performance of color image quantization using MLSDE, its results are compared with other famous color image quantization technique. However, the simulations were run for 10 times and the outcomes are illustrated as averages and standard deviations. The results of MLSDE show the ability to converge to the best solution found by the preliminary phase by using a Vbest approach. The MLSDE parameters were initially set as follows: pkmeans = 0.1, s = 20, tmax = 50. The Table 5 shows the comparative result. MSE results of MLSDE illustrate that color quantization using CQ-MLSDE [12] remarkably enhances the quantized image quality in most of the cases.

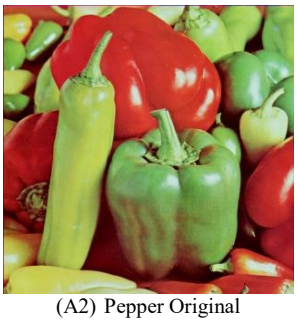
Figure 5.4 Quantization result of images

(A1) Lenna Original

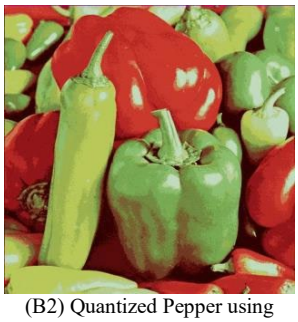
(B1) Quantized Lenna using CQ-MLSDE 16 colors

(C1) Quantized Lenna using CQ-MLSDE 32 colors

(D1) Quantized Lenna using CQ-MLSDE 64 colors



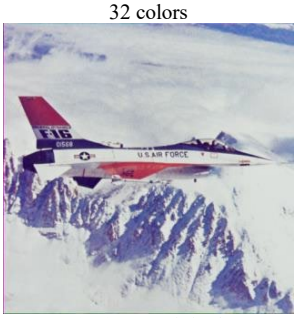
(A2) Pepper Original



(B2) Quantized Pepper using CQ-MLSDE 16 colors

(C2) Quantized Pepper using CQ-MLSDE 32 colors

(D2) Quantized Pepper using CQ-MLSDE 64 colors

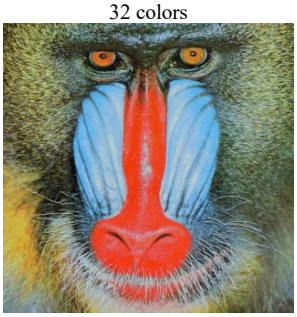


(A3) Jet Original

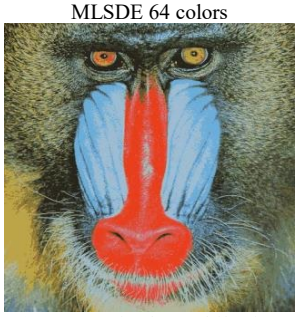
(B3) Quantized Jet using CQ-MLSDE 16 colors

(C3) Quantized Jet using CQ-MLSDE 32 colors

(D3) Quantized Jet using CQ-MLSDE 64 colors

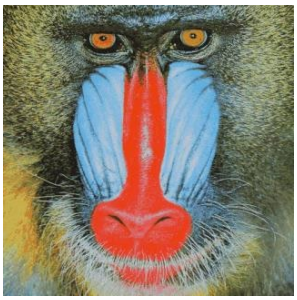


(A4) Mandril Original

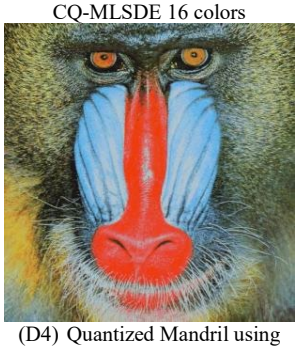


(B4) Quantized Mandril using

CQ-MLSDE 16 colors



(C4) Quantized Mandril using CQ-MLSDE 32 colors



(D4) Quantized Mandril using CQ-MLSDE 64 colors

image	K	SOM	GCMA	PSO	CQ-MLSDE
Lenna	16	235.6	332	210.203	201.894
	32	126.40	179	119.167	112.1360
	64	74.700	113	77.846	72.107
Peppers	16	425.60	471	399.36	387.6023
	32	244.50	263	232.046	229.2507
	64	141.60	148	137.322	138.463
Jet	16	121.70	199	122.867	113.506
	32	65.000	96	71.564	43.500
	64	38.100	54	56.339	40.421
Mandril	16	629.00	606	630.975	519.4218
	32	373.60	348	375.933	269.851
	64	234.00	213	237.331	139.1367

Table 5.1 The MSDE was tested by picking number of clusters 16, 32, 64 for lenna, pepper, jet and mandrill images. Fig. 1 to 4 demonstrate the quantization of lenna, pepper, jet and mandrill images.

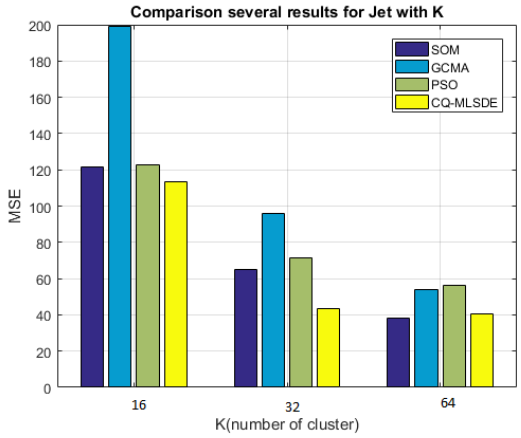


Figure 5.5 Experimental results Jet

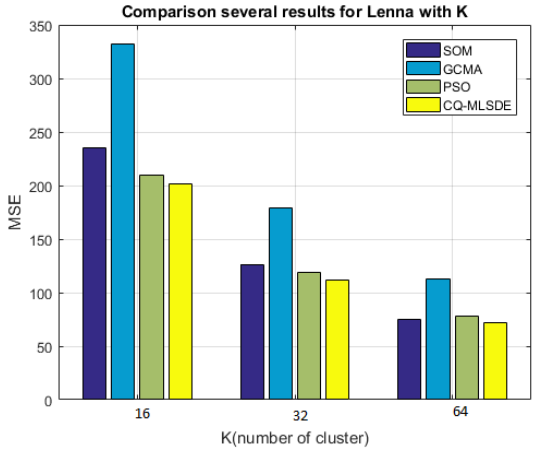


Figure 5.6 Experimental results Lenna

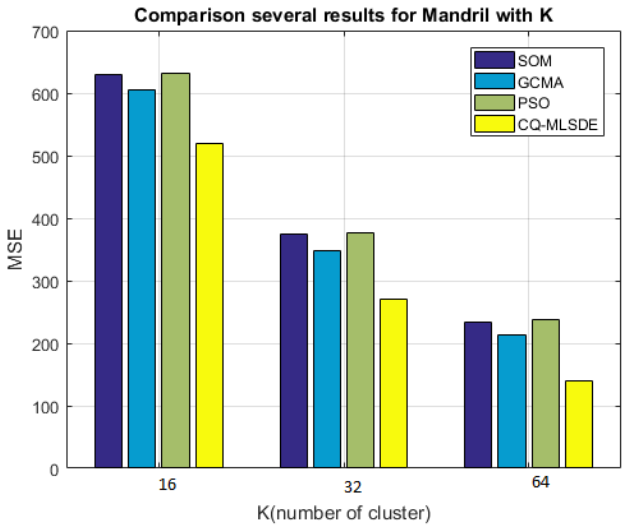


Figure 5.7 Experimental results Mandril

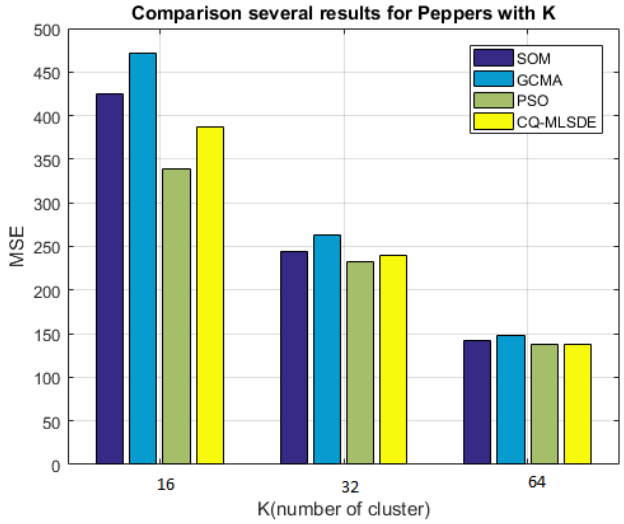


Figure 5.8 Experimental results Peppers

CHAPTER 6: CONCLUSIONS

A multi-layer strategy for DE, MLSDE has been proposed in this paper. Due to the multi-strategy approach, the diversity of the offspring can be preserved. MLSDE involves three mutation and crossover strategies to produce three different trail vector generations. These three strategies provide many advantages, such as diversity and ability to search around promise area and can therefore complement each other. MLSDE depends entirely on these strategies; thus, the parameters are fixed during the evolution process. The experimental of 16 global numerical optimization problems showed that the operations of MLSDE are more efficient and effective than those other algorithms. The performance of the MLSDE algorithm was validated over a set of 16 benchmark functions. The experimental results demonstrate that the multi-layer strategies approach is successful in maintaining population diversity. MLSDE not only performed better than the JADE, JDE, and SADE for most of functions but was also competitive and it reach near global optima. The proposed MLSDE can get more improvements such as it can be employed with other distributions such as e Cauchy distributions that have shown promising results in evolutionary algorithms.

REFERENCES

[1] S. Das and P. N. Suganthan, "Differential evolution: a survey of the state-of-the-art," IEEE Trans. Evol. Comput., vol. 15, pp. 4-31, 2011.

[2] S. S. Rao and S. S. Rao, Engineering Optimization: Theory and Practice. Hoboken, NJ: John Wiley & Sons, 2009.

[3] K. Price, R. M. Storn, and J. A. Lampinen, Differential Evolution: A Practical Approach to Global Optimization. Berlin: Springer Science & Business Media, 2006.

[4] J. Brownlee, Clever algorithms: nature-inspired programming recipes: Jason Brownlee, 2011.

[5] J. Zhang and A. C. Sanderson, Adaptive differential evolution: Springer, 2009.

[6] V. Feoktistov, Differential Evolution. Dordrecht: Springer, 2006.

[7] R. Storn, "On the usage of differential evolution for function optimization," in NAFIPS, 1996 Biennial Conference of the North American Fuzzy Information Processing Society, 1996, 1996, pp. 519-523.

[8] K. V. Price, R. M. Storn, and J. A. Lampinen, "The differential evolution algorithm," in Differential Evolution: A Practical Approach to Global Optimization, K. V. Price, R. M. Storn, and J. A. Lampinen, Eds., ed Berlin, Heidelberg: Springer, 2005, pp. 37-134.

[9] S.-M. Guo, C.-C. Yang, P.-H. Hsu, and J. S.-H. Tsai, "Improving differential evolution with a successful-parent-selecting framework," IEEE Trans. Evol. Comput., vol. 19, pp. 717-730, 2015.

[10] Á. E. Eiben, R. Hinterding, and Z. Michalewicz, "Parameter control in evolutionary algorithms," IEEE Trans. Evol. Comput., vol. 3, pp. 124-141, 1999.

[11] T. Kok, B. AE, E. JN, H. Spaink, C. Kari, G. T, et al., Natural Computing Series. Berlin: Springer, 2006.

[12] D. H. Wolpert and W. G. Macready, "The mathematics of search," Technical Report SFI-TR-95-02-010, Santa Fe Institute1995.

[13] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," IEEE Trans. Evol. Comput., vol. 1, 1997, pp. 67-82.

[14] D. Zaharie, "On the explorative power of differential evolution," in 3rd International Workshop on Symbolic and Numerical Algorithms on Scientific Computing, SYNASC-2001, Timişoara, Romania, 2001.

[15] J. Liu, "On setting the control parameter of the differential evolution method," in Proceedings of the 8th international conference on soft computing (MENDEL 2002), 2002, pp. 11-18.

[16] T. Šmuc, "Improving convergence properties of the differential evolution algorithm," in MENDEL 2002-8th International Conference on Soft Computing, 2002.

[17] I. K. Yalcin and M. Gokmen, "Integrating differential evolution and condensation algorithms for license plate tracking," in Pattern Recognition, 2006. ICPR 2006. 18th International Conference on, 2006, pp. 658-661.

[18] J. Zhang and A. C. Sanderson, "JADE: adaptive differential evolution with optional external archive," IEEE Transactions on evolutionary computation, vol. 13, 2009, pp. 945-958.

[19] Á. Baíllo, M. Ventosa, M. Rivier, and A. Ramos, "Strategic bidding in a competitive electricity market: a decomposition approach," in Power Tech Proceedings, 2001 IEEE Porto, vol. 1, 2001, pp. 6.

[20] J. H. Van Sickel, K. Y. Lee, and J. S. Heo, "Differential evolution and its applications to power plant control," in Intelligent Systems Applications to Power Systems, 2007. ISAP 2007. International Conference on, 2007, pp. 1-6.

[21] X. Wang, H. Cheng, and M. Huang, "QoS multicast routing protocol oriented to cognitive network using competitive coevolutionary algorithm," Expert Systems with Applications, vol. 41, 2014, pp. 4513-4528.

[22] A. A. El Ela, M. Abido, and S. Spea, "Optimal power flow using differential evolution algorithm," Electric Power Systems Research, vol. 80, , 2010, pp. 878-885.

[23] J. C. Goswami, R. Mydur, and P. Wu, "Application of differential evolution algorithm to model-based well log-data inversion," in Antennas and Propagation Society International Symposium, 2002. IEEE, 2002, pp. 318-321.

[24] C.-T. Su and C.-S. Lee, "Network reconfiguration of distribution systems using improved mixed-integer hybrid differential evolution," IEEE Transactions on Power Delivery, vol. 18, , 2003,pp. 1022-1027.

[25] Y. Boughari, G. Ghazi, R. M. Botez, and F. Theel, "New Methodology for Optimal Flight Control Using Differential Evolution Algorithms Applied on the

Cessna Citation X Business Aircraft–Part 1. Design and Optimization," INCAS Bulletin, vol. 9, , 2017, p. 31.

[26] K. V. Price, "Differential evolution vs. the functions of the 2/sup nd/ ICEO," in IEEE International Conference on Evolutionary Computation, 1997, pp. 153-157.

[27] F. Xue, A. C. Sanderson, P. P. Bonissone, and R. J. Graves, "Fuzzy logic controlled multi-objective differential evolution," in The 14th IEEE International Conference on Fuzzy Systems, 2005. FUZZ'05, 2005, pp. 720-725.

[28] R. Storn, "Differrential evolution-a simple and efficient adaptive scheme for global optimization over continuous spaces," Technical Report, International Computer Science Institute, 1995.

[29] E. Mezura-Montes, J. Velázquez-Reyes, and C. A. Coello Coello, "A comparative study of differential evolution variants for global optimization," in Proceedings of the 8th annual conference on Genetic and evolutionary computation, 2006, pp. 485-492.

[30] J. Vesterstrom and R. Thomsen, "A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems," in Congress on Evolutionary Computation, 2004. CEC2004, 2004, pp. 1980-1987.

[31] I. Fister, M. Mernik, and J. Brest, "Hybridization of Evolutionary Algorithms," arXiv preprint arXiv:1301.0929, 2013.

[32] H. Chunping and Y. Xuefeng, "An immune self-adaptive differential evolution algorithm with application to estimate kinetic parameters for homogeneous mercury oxidation," Chin. J. Chem. Eng., vol. 17, , 2009,pp. 232-240.

[33] J. Ilonen, J.-K. Kamarainen, and J. Lampinen, "Differential evolution training algorithm for feed-forward neural networks," Neural Process. Lett., vol. 17, 2003, pp. 93-105.

[34] G. Eiben and M. C. Schut, "New ways to calibrate evolutionary algorithms," in Advances in Metaheuristics for Hard Optimization, P. Siarry and Z. Michalewicz, Eds., ed Berlin, Heidelberg: Springer, 2007, pp. 153-177.

[35] P. J. Angeline, "Adaptive and self-adaptive evolutionary computations," in Computational Intelligence: A Dynamic Systems Perspective, 1995.

[36] A. E. Eiben and J. E. Smith, Introduction to evolutionary computing vol. 53: Springer, 2003.

[37] J. Liu, J. Lampinen, R. Matousek, and P. Osmera, "Adaptive parameter control of differential evolution," in Proc. of MENDEL, 2002, pp. 19-26.

[38] H.-Y. Fan and J. Lampinen, "A trigonometric mutation operation to differential evolution," J. Glob. Optimiz., vol. 27, 2003, pp. 105-129.

[39] S. Das, A. Abraham, U. K. Chakraborty, and A. Konar, "Differential evolution using a neighborhood-based mutation operator," IEEE Trans. Evol. Comput., vol. 13, , 2009, pp. 526-553.

[40] A. Qing, Differential evolution: fundamentals and applications in electrical engineering: John Wiley & Sons, 2009.

[41] C. Lin, A. Qing, and Q. Feng, "A comparative study of crossover in differential evolution," J. Heurist., vol. 17, 2011 pp. 675-703.

[42] H. A. Abbass, "The self-adaptive pareto differential evolution algorithm," in Proceedings of the 2002 Congress on Evolutionary Computation, 2002. CEC'02, 2002, pp. 831-836.

[43] J. Teo, "Exploring dynamic self-adaptive populations in differential evolution," Soft Comput., vol. 10, 2006, pp. 673-686.

[44] J. Liu and J. Lampinen, "A fuzzy adaptive differential evolution algorithm," Soft Comput., vol. 9, 2005, pp. 448-462.

[45] A. K. Qin and P. N. Suganthan, "Self-adaptive differential evolution algorithm for numerical optimization," in The 2005 IEEE Congress on Evolutionary Computation, 2005, pp. 1785-1791.

[46] J. Brest, S. Greiner, B. Boskovic, M. Mernik, and V. Zumer, "Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems," IEEE Trans. Evol. Comput., vol. 10, , 2006, pp. 646-657.

[47] P. Kaelo and M. Ali, "Differential evolution algorithms using hybrid mutation," Comput. Optimiz. Appl., vol. 37, pp. 231-246, 2007.

[48] A. M. A. Rocha and E. M. d. G. Fernandes, "On charge effects to the electromagnetism-like algorithm," in 20th EURO Mini Conference:" Continuous Optimization and Knowledge-Based Technologies", 2008, pp. 198-203.

[49] Y. Wang, Z. Cai, and Q. Zhang, "Differential evolution with composite trial vector generation strategies and control parameters," IEEE Trans. Evol. Comput., vol. 15, , 2011, pp. 55-66.

[50] D. Zaharie, "Control of population diversity and adaptation in differential evolution algorithms," in Proc. of MENDEL, 2003, pp. 41-46.

[51] P. Kumar and M. Pant, "A self adaptive differential evolution algorithm for global optimization," in International Conference on Swarm, Evolutionary, and Memetic Computing, 2010, pp. 103-110.

[52] J. Brest, B. Bošković, S. Greiner, V. Žumer, and M. S. Maučec, "Performance comparison of self-adaptive and adaptive differential evolution algorithms," Soft Comput., vol. 11, 2007, pp. 617-629.

[53] M. Shojafar, C. Canali, R. Lancellotti, and J. Abawajy, "Adaptive computing-plus-communication optimization framework for multimedia processing in cloud systems," IEEE Transactions on Cloud Computing, 2016.

[54] M. Shojafar, N. Cordeschi, and E. Baccarelli, "Energy-efficient adaptive resource management for real-time vehicular cloud services," IEEE Transactions on Cloud computing, 2016.

[55] T. Hendtlass, "A combined swarm differential evolution algorithm for optimization problems," in International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, 2001, pp. 11-18.

[56] M. Shojafar, S. Javanmardi, S. Abolfazli, and N. Cordeschi, "FUGE: A joint meta-heuristic approach to cloud job scheduling algorithm using fuzzy theory and a genetic method," Cluster Computing, vol. 18, , 2015, pp. 829-844.

[57] Z. Yang, K. Tang, and X. Yao, "Differential evolution for high-dimensional function optimization," in IEEE Congress on Evolutionary Computation, 2007. CEC 2007, pp. 3523-3530.

[58] M.-X. Ling, F.-Y. Wang, X. Ding, Y.-H. Hu, J.-B. Zhou, R. E. Zartman, et al., "Cretaceous ridge subduction along the lower Yangtze River belt, eastern China," Econ. Geol., vol. 104, , 2009, pp. 303-321.

[59] B. Babu and R. Angira, "Modified differential evolution (MDE) for optimization of non-linear chemical processes," Comput. Chem. Eng., vol. 30, , 2006, pp. 989-1002.

[60] S. Sayah and K. Zehar, "Modified differential evolution algorithm for optimal power flow with non-smooth cost functions," Energy Convers. Manag., vol. 49, , 2008,pp. 3036-3042.

[61] L. Lakshminarasimman and S. Subramanian, "Short-term scheduling of hydrothermal power system with cascaded reservoirs by using modified differential evolution," IEE Proc.-Gener. Transm. Distrib., vol. 153, 2006,pp. 693-700.

[62] S. M. Islam, S. Das, S. Ghosh, S. Roy, and P. N. Suganthan, "An adaptive differential evolution algorithm with novel mutation and crossover strategies for global numerical optimization," IEEE Trans. Syst., Man Cybern., Part B: Cybern., vol. 42, , 2012, pp. 482-500.

[63] Y. Cai and J. Wang, "Differential evolution with neighborhood and direction information for numerical optimization," IEEE Trans. Cybern., vol. 43, , 2013,pp. 2202-2215.

[64] R. M. Alguliev, R. M. Aliguliyev, and N. R. Isazade, "DESAMC+ DocSum: differential evolution with self-adaptive mutation and crossover parameters for multi-document summarization," Knowl. -Based Syst., vol. 36, 2012, pp. 21-38.

[65] A. Selamat, N. T. Nguyen, and H. Haron, Intelligent Information and Database Systems: 5th Asian Conference, ACIIDS 2013, Kuala Lumpur, Malaysia, March 18-20, 2013, Proceedings vol. 7803: Springer, 2013.

[66] J. Zhang and A. C. Sanderson, "JADE: adaptive differential evolution with optional external archive," IEEE Trans. Evol. Comput., vol. 13, pp. 945-958, 2009.

[67] J. Zhang and A. C. Sanderson, "JADE: Self-adaptive differential evolution with fast and reliable convergence performance," in IEEE Congress on Evolutionary Computation, 2007. CEC 2007, 2007, pp. 2251-2258.

[68] R. Tanabe and A. Fukunaga, "Success-history based parameter adaptation for differential evolution," in 2013 IEEE Congress on Evolutionary Computation (CEC), 2013, pp. 71-78.

[69] S. Ghosh, S. Das, S. Roy, S. M. Islam, and P. N. Suganthan, "A differential covariance matrix adaptation evolutionary algorithm for real parameter optimization," Inf. Sci., vol. 182, , 2012, pp. 199-219.

[70] S. Ghosh, S. Roy, S. M. Islam, S. Das, and P. N. Suganthan, "A differential covariance matrix adaptation evolutionary algorithm for global optimization," in 2011 IEEE Symposium on Differential Evolution (SDE), 2011, pp. 1-8.

[71] S. M. Elsayed, R. A. Sarker, and D. L. Essam, "An improved self-adaptive differential evolution algorithm for optimization problems," IEEE Trans. Ind. Inform., vol. 9, , 2013, pp. 89-99.

[72] D. Lichtblau, "Relative position indexing approach," in Differential Evolution: A Handbook for Global Permutation-Based Combinatorial Optimization, ed: Springer, 2009, pp. 81-120.

[73] C. Blum, J. Puchinger, G. R. Raidl, and A. Roli, "Hybrid metaheuristics in combinatorial optimization: a survey," Appl. Soft Comput., vol. 11, 2011, pp. 4135-4151.

[74] E.-N. Dragoi and V. Dafinescu, "Parameter control and hybridization techniques in differential evolution: a survey," Artif. Intell. Rev., vol. 45, pp. 447-470, 2016.

[75] D. E. Goldberg and H. John, "Holland. Genetic algorithms and machine learning," Machine learning, vol. 3, 1988, pp. 95-99.

[76] P. Vas, Artificial-intelligence-based electrical machines and drives: application of fuzzy, neural, fuzzy-neural, and genetic-algorithm-based techniques vol. 45: Oxford university press, 1999.

[77] B. K. Panigrahi, P. N. Suganthan, S. Das, and S. S. Dash, "Swarm, Evolutionary, and Memetic Computing," in Third International Conference SEMCCO, 2010.

[78] E. Nwankwor, A. K. Nagar, and D. Reid, "Hybrid differential evolution and particle swarm optimization for optimal well placement," Comput. Geosci., vol. 17, , 2013, pp. 249-268.

[79] F. Vitaliy, Differential Evolution–In Search of Solutions. New York: Springer, 2006.

[80] I. Fister and I. Fister Jr, Adaptation and Hybridization in Computational Intelligence vol. 18. Cham: Springer, 2015.

[81] R. Thangaraj, M. Pant, A. Abraham, and P. Bouvry, "Particle swarm optimization: hybridization perspectives and experimental illustrations," Appl. Math. Comput., vol. 217, , 2011,pp. 5208-5226.

[82] R. Eberhart and J. Kennedy, "Particle swarm optimization," in Proceeding of IEEE International Conference on Neural Network, Perth, Australia, 1995, pp. 1942-1948.

[83] Y. Shi and R. C. Eberhart, "Empirical study of particle swarm optimization," in Proceedings of the 1999 Congress on Evolutionary Computation, 1999. CEC 99, 1999, pp. 1945-1950.

[84] I. C. Trelea, "The particle swarm optimization algorithm: convergence analysis and parameter selection," Inf. Process. Lett., vol. 85, , 2003, pp. 317-325.

[85] J. Kennedy and R. Mendes, "Population structure and particle swarm performance," in Proceedings of the 2002 Congress on Evolutionary Computation, 2002. CEC'02, 2002, pp. 1671-1676.

[86] H. Zhenya, W. Chengjian, Y. Luxi, G. Xiqi, Y. Susu, R. C. Eberhart, et al., "Extracting rules from fuzzy neural network by particle swarm optimisation," in The 1998 IEEE International Conference on Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence, 1998, pp. 74-77.

[87] S. Das, A. Abraham, and A. Konar, "Particle swarm optimization and differential evolution algorithms: technical analysis, applications and hybridization perspectives," in Advances of computational intelligence in industrial systems, ed: Springer, 2008, pp. 1-38.

[88] J. H. Van Sickel, K. Y. Lee, and J. S. Heo, "Differential evolution and its applications to power plant control," in International Conference on Intelligent Systems Applications to Power Systems, 2007. ISAP 2007, pp. 1-6.

[89] X. Yu, J. Cao, H. Shan, L. Zhu, and J. Guo, "An adaptive hybrid algorithm based on particle swarm optimization and differential evolution for global optimization," ScientificWorldJournal, vol. 2014, 2014, p. 215472.

[90] W.-J. Zhang and X.-F. Xie, "DEPSO: hybrid particle swarm with differential evolution operator," in IEEE International Conference on Systems, Man and Cybernetics, 2003, pp. 3816-3821.

[91] B. Liu, "Uncertain risk analysis and uncertain reliability analysis," J. Uncertain Syst., vol. 4, , 2010, pp. 163-170.

[92] H. Liu, Z. Cai, and Y. Wang, "Hybridizing particle swarm optimization with differential evolution for constrained numerical and engineering optimization," Appl. Soft Comput., vol. 10, 2010, pp. 629-640.

[93] J. Hästbacka, A. de la Chapelle, M. M. Mahtani, G. Clines, M. P. Reeve-Daly, M. Daly, et al., "The diastrophic dysplasia gene encodes a novel sulfate transporter: positional cloning by fine-structure linkage disequilibrium mapping," Cell, vol. 78, , 1994, pp. 1073-1087.

[94] S. Das, A. Abraham, and A. Konar, "Automatic clustering using an improved differential evolution algorithm," IEEE Trans. Syst., Man Cybern., Part A: Syst. Humans, vol. 38, 2008,pp. 218-237.

[95] D. E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning. USA: Addison, 1989.

[96] E. Mezura-Montes, "Nature-Inspired Algorithms Evolutionary and Swarm Intelligence Approaches," A Tutorial in MICAI, vol. 2008, 2008.

[97] X. Xu and Y. Li, "Comparison between particle swarm optimization, differential evolution and multi-parents crossover," in 2007 International Conference on Computational Intelligence and Security, 2007, pp. 124-127.

[98] I. Codreanu, "A parallel between differential evolution and genetic algorithms with exemplification in a microfluidics optimization problem," in Semiconductor Conference, 2005. CAS 2005 Proceedings. 2005 International, 2005, pp. 421-424.

[99] M. R. Sentinella, "Comparison and integrated use of differential evolution and genetic algorithms for space trajectory optimisation," in IEEE Congress on Evolutionary Computation, 2007. CEC 2007, 2007, pp. 973-978.

[100] B. Hegerty, C.-C. Hung, and K. Kasprak, "A comparative study on differential evolution and genetic algorithms for some combinatorial problems," in Proceedings of 8th Mexican International Conference on Artificial Intelligence, 2009, pp. 9-13.

[101] T. W. Liao, "Two hybrid differential evolution algorithms for engineering design optimization," Appl. Soft Comput., vol. 10, , 2010 , pp. 1188-1199.

[102] I. Boussaïd, A. Chatterjee, P. Siarry, and M. Ahmed-Nacer, "Two-stage update biogeography-based optimization using differential evolution algorithm (DBBO)," Comput. Oper. Res., vol. 38, , 2011, pp. 1188-1198.

[103] I. Boussaïd, A. Chatterjee, P. Siarry, and M. Ahmed-Nacer, "Hybridizing biogeography-based optimization with differential evolution for optimal power allocation in wireless sensor networks," IEEE Trans. Veh. Technol., vol. 60, , 2011, pp. 2347-2353.

[104] R. Moral, D. Sahoo, and G. Dulikravich, "Multi-objective hybrid evolutionary optimization with automatic switching," in 11th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, 2006, p. 6976.

[105] H. Guo, Y. Li, J. Li, H. Sun, D. Wang, and X. Chen, "Differential evolution improved with self-adaptive control parameters based on simulated annealing," Swarm Evol. Comput., vol. 19, , 2014, pp. 52-67.

[106] N. Pholdee, S. Bureerat, and A. R. Yıldız, "Hybrid real-code population-based incremental learning and differential evolution for many-objective optimisation of an automotive floor-frame," Int. J. Veh. Design, vol. 73, 2017, pp. 20-53.

[107] N. Pholdee and S. Bureerat, "Hybridisation of real-code population-based incremental learning and differential evolution for multiobjective design of trusses," Inf. Sci., vol. 223, 2013, pp. 136-152.

[108] N. Pholdee and S. Bureerat, "Hybrid real-code population-based incremental learning and approximate gradients for multi-objective truss design," Eng. Optim., vol. 46 , 2014, pp. 1032-1051.

[109] S. Bureerat, N. Pholdee, W.-W. Park, and D.-K. Kim, "An Improved Teaching-Learning Based Optimization for Optimization of Flatness of a Strip During a Coiling Process," in International Workshop on Multi-disciplinary Trends in Artificial Intelligence, 2016, pp. 12-23.

[110] Neri et al.

[111] F. Neri and C. Cotta, "Memetic algorithms and memetic computing optimization: a literature review," Swarm Evol. Comput., vol. 2, 2012, pp. 1-14.

[112] D. Zou, J. Wu, L. Gao, and S. Li, "A modified differential evolution algorithm for unconstrained optimization problems," Neurocomputing, vol. 120, , 2013 , pp. 469-481.

[113] Z.-h. Zhan, and Jun Zhang, "Enhance differential evolution with random walk," in ACM Proceedings of the 14th Annual Conference Companion on Genetic and Evolutionary Computation, 2012, pp. 1513-1514.

[114] W.-j. Yu and J. Zhang, "Multi-population differential evolution with adaptive parameter control for global optimization," in Proceedings of the 13th annual conference on Genetic and evolutionary computation, 2011, pp. 1093-1098.

[115] P. Bujok, J. Tvrdik, and R. Polakova, "Differential evolution with rotation-invariant mutation and competing-strategies adaptation," in Evolutionary Computation (CEC), 2014 IEEE Congress on, 2014, pp. 2253-2258.

[116] S. M. Islam, S. Das, S. Ghosh, S. Roy, and P. N. Suganthan, "An adaptive differential evolution algorithm with novel mutation and crossover strategies for global numerical optimization," IEEE Transactions on Systems, Man and Cybernetics, Part B: Cybernetics, vol. 42, 2012, pp. 482-500.

[117] A. Trivedi, D. Srinivasan, S. Biswas, and T. Reindl, "A genetic algorithm–differential evolution-based hybrid framework: case study on unit commitment scheduling problem," Information Sciences, vol. 354, 2016, pp. 275-300.

[118] M. Pant, R. Thangaraj, C. Grosan, and A. Abraham, "Hybrid differential evolution-particle swarm optimization algorithm for solving global optimization problems," in Digital Information Management, 2008. ICDIM 2008. Third International Conference on, 2008, pp. 18-24.

[119] A. Antoniou and W.-S. Lu, The Optimization Problem: Springer, 2007.

[120] B. Babu and S. Munawar, "Differential evolution strategies for optimal design of shell-and-tube heat exchangers," Chemical Engineering Science, vol. 62 , 2007, pp. 3720-3739.

[121] F. Neri and V. Tirronen, "Recent advances in differential evolution: a survey and experimental analysis," Artificial Intelligence Review, vol. 33, 2010, pp. 61-106.

[122] S. Das and P. N. Suganthan, "Differential evolution: a survey of the state-of-the-art," IEEE transactions on evolutionary computation, vol. 15, 2011, pp. 4-31.

[123] C. Lin, A. Qing, and Q. Feng, "A comparative study of crossover in differential evolution," Journal of Heuristics, vol. 17, 2011, pp. 675-703.

[124] S.-M. Guo, C.-C. Yang, P.-H. Hsu, and J. S.-H. Tsai, "Improving differential evolution with a successful-parent-selecting framework," IEEE Transactions on Evolutionary Computation, vol. 19, , 2015, pp. 717-730.

[125] Z. Peng, J. Liao, and Y. Cai, "Differential evolution with distributed direction information based mutation operators: an optimization technique for big data," Journal of Ambient Intelligence and Humanized Computing, vol. 6, 2015, pp. 481-494.

[126] M. Yang, C. Li, Z. Cai, and J. Guan, "Differential evolution with auto-enhanced population diversity," IEEE transactions on cybernetics, vol. 45 , 2015, pp. 302-315.

[127] G. C. Onwubolu and D. Davendra, Differential evolution: a handbook for global permutation-based combinatorial optimization vol. 175: Springer Science & Business Media, 2009.

[128] A. K. Qin, V. L. Huang, and P. N. Suganthan, "Differential evolution algorithm with strategy adaptation for global numerical optimization," IEEE transactions on Evolutionary Computation, vol. 13 , 2009,pp. 398-417.

[129] T.-T. Chang and H.-C. Chang, "Application of differential evolution to passive shunt harmonic filter planning," in Harmonics and Quality of Power Proceedings, 1998. Proceedings. 8th International Conference On, 1998, pp. 149-153.

[130] Y. Wang, Z. Cai, and Q. Zhang, "Differential evolution with composite trial vector generation strategies and control parameters," IEEE Transactions on Evolutionary Computation, vol. 15, 2011, pp. 55-66.

[131] M. Ali, M. Pant, and A. Abraham, "Improved differential evolution algorithm with decentralisation of population," International Journal of Bio-Inspired Computation, vol. 3, 2011, pp. 17-30.

[132] R. A. Sarker, S. M. Elsayed, and T. Ray, "Differential Evolution With Dynamic Parameters Selection for Optimization Problems," IEEE Trans. Evolutionary Computation, vol. 18, 2014, pp. 689-707.

[133] J. Wang, J. Liao, Y. Zhou, and Y. Cai, "Differential evolution enhanced with multiobjective sorting-based mutation operators," IEEE transactions on cybernetics, vol. 44, 2014, pp. 2792-2805.

[134] Y. Wang, H.-X. Li, T. Huang, and L. Li, "Differential evolution based on covariance matrix learning and bimodal distribution parameter setting," Applied Soft Computing, vol. 18, 2014, pp. 232-247.

[135] W.-J. Yu, M. Shen, W.-N. Chen, Z.-H. Zhan, Y.-J. Gong, Y. Lin, et al., "Differential evolution with two-level parameter adaptation," IEEE Transactions on Cybernetics, vol. 44, , 2014, pp. 1080-1099.

[136] W. M. Spears and K. A. Jong, The role of mutation and recombination in evolutionary algorithms: George Mason University Fairfax, VA, 1998.

[137] R. Storn, "Differential evolution research–trends and open questions," in Advances in differential evolution, ed: Springer, 2008, pp. 1-31.

[138] H.-Y. Fan and J. Lampinen, "A trigonometric mutation operation to differential evolution," Journal of global optimization, vol. 27 , 2003, pp. 105-129.

[139] S. Kukkonen and J. Lampinen, "An extension of generalized differential evolution for multi-objective optimization with constraints," in International Conference on Parallel Problem Solving from Nature, 2004, pp. 752-761.

[140] V. Feoktistov, Differential evolution: in search of solutions vol. 5: Springer Science & Business Media, 2007.

[141] P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y.-P. Chen, A. Auger, et al., "Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization," KanGAL report, vol. 2005005, 2005, p. 2005.

[142] J. Brest, S. Greiner, B. Boskovic, M. Mernik, and V. Zumer, "Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems," IEEE transactions on evolutionary computation, vol. 10, 2006, pp. 646-657.

[143] R. Gämperle, S. D. Müller, and P. Koumoutsakos, "A parameter study for differential evolution," Advances in intelligent systems, fuzzy systems, evolutionary computation, vol. 10, 2002, pp. 293-298.